DISS. ETH NO. 27739

Robust Visual Odometry

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH Zürich

(Dr. sc. ETH Zürich)

presented by

Peidong Liu

Master of Engineering
National University of Singapore

Born Oct 11, 1988
Citizen of Republic of China

accepted on the recommendation of

Examiner
Prof. Marc Pollefeys

Co-Examiners
Prof. Andreas Geiger
Prof. Davide Scaramuzza

2021

Abstract

Visual odometry (VO) is a technique used to estimate the camera motion from acquired images. As a fundamental block for many real-world applications, such as robotics and virtual/mixed/augmented reality, great progress has been made during the last decades. There have been many algorithms being proposed in the literature: ranging from classical geometric approaches, deep learning based approaches to hybrid approaches. Those pipelines usually can achieve reasonable good performance in controlled environments. While many state-of-the-art algorithms have been proposed, robustness is still a remaining challenge for visual odometry methods. Poor lighting conditions, motion blur, poor textured environment et al. can still fail visual odometry methods easily, which would limit their deployments to the challenging real-world environments.

In this dissertation, we propose methods to improve the robustness of visual odometry algorithms from three different perspectives: 1) Due to the low contrast of images captured at low-lighting conditions, visual odometry methods would have difficulties to detect enough salient feature points for motion estimation. One solution is to enlarge the field of view of the camera, so that more salient feature points can be detected. We therefore propose a visual odometry algorithm for a multi-camera rig to cover 360° surrounding scene. Experimental results demonstrate that it improves both the accuracy and robustness of visual odometry algorithm at night conditions. 2) Instead of enlarging the field of view, we try to further improve the robustness by enhancing the quality of input images via a deep convolutional neural network. The enhanced images can then be fed into a normal visual odometry pipeline for improved performance. Both motion blur and rolling shutter effect are common artifacts that degrade image qualities. Motion blur usually occurs in low-light conditions where longer exposure times are necessary. Rolling shutter effect occurs in images

captured by a CMOS camera while the camera is moving. It distorts the captured images, which would degrade the performance of normal VO methods. We thus propose deep neural networks to deblur the input images and remove the rolling shutter distortions, so that the performance of vision-based algorithms (e.g. visual odometry) can be improved. 3) To further improve the robustness of visual odomtry against motion blurred images, we propose a novel hybrid visual odometry method, which models the image formation process of motion blur. This allows us to explicitly model the motion blur in the image and leverage it for tracking. In experiments we show that by directly modeling the image formation process, we are able to improve robustness of the visual odometry, while keeping comparable accuracy as that for images without motion blur.

We also study the limitations of existing learning based methods, which aim to improve both the accuracy and robustness of VO methods by using deep networks, in this dissertation. By demonstrating the fact that even a single image motion estimation network can predict reasonable camera motions on KITTI dataset [GLU12], which is commonly used by the community, we conclude that the KITTI dataset might not be sufficient for those learning based approaches to evaluate on. The reason is that a single image from KITTI dataset already contains many semantic motion cues for the network to exploit for motion prediction.

Abstrakt

Visuelle Odometrie ist eine Technik, mit der die Kamerabewegung aus aufgenommenen Bildern geschtzt wird. Es dient als grundlegender Block fr viele reale Anwendungen wie Robotik und virtuelle/gemischte/erweiterte Realitt und in den letzten Jahrzehnten wurden in diesem Gebiet groe Fortschritte erzielt. In der Literatur wurden viele Algorithmen vorgeschlagen: von klassischen geometrischen Anstzen ber Deep-Learning-basierte Anstze bis hin zu hybriden Anstzen. Diese Pipelines knnen normalerweise in kontrollierten Umgebungen eine angemessen gute Leistung erzielen. Obwohl bereits viele Algorithmen nach dem letzten Stand der Technik vorgeschlagen wurden, bleibt die Robustheit eine Herausforderung fr visuelle Odometrie-Verfahren. Wegen schlechten Lichtverhltnissen, Bewegungsunschrfen und schlecht strukturierten Umgebungen knnen visuelle Odometrie-Methoden schnell scheitern, was ihre Anwendbarkeit auf realen Umgebungen einschrnkt.

In dieser Dissertation schlagen wir daher Methoden vor, welche die Robustheit von visuellen Odometrie-Algorithmen aus drei verschiedenen Perspektiven verbessert: 1) Aufgrund des geringen Kontrasts von Bildern, die bei schlechten lichtverhltnissen aufgenommen wurden, haben visuelle Odometrie-Methoden Schwierigkeiten, gengend hervorstechende Merkmalspunkte fr die Bewegungsschtzung zu erkennen. Eine Lsung besteht darin, das Sichtfeld der Kamera zu vergrern, damit mehr hervorstechendere Merkmalspunkte erkannt werden knnen. Wir schlagen daher einen visuellen Odometrie-Algorithmus fr ein Multi-Kamera-Rig vor, um die 360° umgebende Szene abzudecken. Experimentelle Ergebnisse zeigen, dass es sowohl die Genauigkeit als auch die Robustheit des visuellen Odometrie-Algorithmus bei Nachtbedingungen verbessert. 2) Anstatt das Sichtfeld zu vergrern, versuchen wir, die Robustheit weiter zu verbessern, indem wir die Qualitt der Eingabebilder ber ein tiefes Convolutional Neural Network

verbessern. Die verbesserten Bilder knnen dann in eine normale visuelle Odometry-Pipeline eingespeist werden, was zur Leistungsverbesserungen fhrt. Sowohl Bewegungsunschrfe als auch der Rolling-Shutter-Effekt sind hufige Artefakte, die die Bildqualitt beeintrchtigen. Bewegungsunschrfe tritt normalerweise bei schlechten Lichtverhltnissen auf, bei denen lngere Belichtungszeiten erforderlich sind. Der Rolling-Shutter-Effekt tritt bei Bildern auf, die von einer CMOS-Kamera aufgenommen wurden, whrend sich die Kamera bewegt. Es verzerrt die aufgenommenen Bilder, was die Leistung normaler VO-Methoden beeintrchtigen wrde. Wir schlagen daher tiefe neuronale Netze vor, um die Eingabebilder zu verschrfen und die Rolling-Shutter-Verzerrungen zu beseitigen, damit die Leistung von visuellen Algorithmen (z. B. visuelle Odometrie) verbessert werden kann. 3) Um die Robustheit der visuellen Odomtrie gegenber bewegungsunscharfen Bildern weiter zu verbessern, schlagen wir eine neuartige hybride visuelle Odometrie-Methode vor, die den Bilderzeugungsprozess von Bewegungsunschrfe modelliert. Auf diese Weise knnen wir die Bewegungsunschrfe im Bild explizit modellieren und fr die Verfolgung nutzen. In Experimenten zeigen wir, dass wir durch direkte Modellierung des Bilderzeugungsprozesses die Robustheit der visuellen Odometrie verbessern und gleichzeitig eine vergleichbare Genauigkeit wie bei Bildern ohne Bewegungsunschrfe beibehalten knnen.

In dieser Dissertation untersuchen wir auch die Grenzen bestehender lernbasierter Methoden, die darauf abzielen, die Genauigkeit und Robustheit von VO-Methoden durch die Verwendung tiefer Netzwerke zu verbessern. Indem wir die Tatsache demonstrieren, dass selbst ein einziges Bildbewegungsschtzungsnetzwerk vernnftige Kamerabewegungen auf dem von der Community hufig verwendeten KITTI-Datensatz [GLU12] vorhersagen kann, schlieen wir, dass der KITTI-Datensatz fr lernbasierte Anstze mglicherweise nicht ausreicht bewerten am. Der Grund dafr ist, dass ein einzelnes Bild aus dem KITTI-Datensatz bereits viele semantische Bewegungshinweise enthlt, die das Netzwerk fr die Bewegungsvorhersage nutzen kann.

Acknowledgement

I sincerely thank all of you, who gave me support and help during my PhD study in ETH Zürich. First of all, I would like to thank my supervisor, Prof. Marc Pollefeys. Without him, I will not have the opportunity to join ETH Zürich for my PhD study. Marc is always supportive and patient with my research. I am grateful that Marc provides me such a great platform from which I can work together with so many excellent researchers. I would also like to thank Dr. Lionel Heng, who collaborated with me for my first publication. Lionel is so kind that always answers my questions and replies with different suggestions at first time, even he is also very busy. He helped me walk out the confused period during the early stage of my PhD study. Another mentor that I want to thank is Prof. Andreas Geiger. Andreas joined our group as a visiting professor, while Marc was on leave for sabbatical to Microsoft. Andreas taught me many things. One of the most valuable lessons that I learnt from him is his research taste. He told me that we should aim high and tackle problems which would have high impact to the community. I am also grateful for his guidance on how to write good academic papers for top-tier conferences. He is so patient that helped revise my paper together with me, so that I can learn how to write in future. I would also like to thank other collaborators and colleagues: Dr. Torsten Sattler, Dr. Viktor Larsson, Dr. Zhaopeng Cui, Dr. Joel Janai, Dr. Lubor Ladicky, Dr. Martin Oswald, Dr. Thomas Schöps, Mr. Xingxing Zuo, Mr. Marcel Geppert, Mr. Songyou Peng and Mr. Zuoyue Li. Thank you for all the fruitful discussions and encouragements for my research. I am also grateful to those responsible anonymous reviewers, who either rejected or accepted my submissions. It is you who taught me what a good work should be and drive me to improve my work to be better and better. I also appreciate the help from our supporting staff, Mr. Thorsten Steenbock, Mrs. Susanne Keller, Mrs. Danielle Luterbacher and Mrs. Ayse Johannes.

I would also like to thank other colleagues from CVG. It is you who make my PhD life to be interesting. I still remember the time that we hangout together for social Thursday, Super Kondi and Ski retreat. I am also grateful to Prof. Davide Scaramuzza for his interests in my work and agrees to be my external independent examiner.

Lastly, I would like to thank the support and company from my family. Especially the understandings and encouragements from my wife, Mrs. Yali Song. Thank you for your encouragements and comforts when I feel depressed with my research. I would also like to thank my parents, who give me the chance to have the best education since I was a child.

It has been one of the most important and memorable periods in my life, which teaches me how to deal with uncertainties, challenges and failures. It is my luck to have all of you during this special journey. Thank you very much to all of you, who make me become a better person and help me acquire the skills to be an independent researcher.

Contents

Abstract i

Acknowledgement v

1. Introduction 1
- 1.1. A classical VO pipeline . . . 2
- 1.2. Contributions of the dissertation . . . 4
- 1.3. Related work . . . 7
  - 1.3.1. Visual odometry . . . 8
  - 1.3.2. Image motion deblurring . . . 13
  - 1.3.3. Rolling shutter effect removal . . . 14
- 1.4. Overview of the dissertation . . . 16

2. Preliminaries 19
- 2.1. Camera models . . . 19
- 2.2. Exponential map . . . 22
- 2.3. Logarithm map . . . 24
- 2.4. State marginalization . . . 26

I. Hardware Perspective 33

3. Direct Visual Odometry for a Fisheye-Stereo Camera 35
- 3.1. Introduction . . . 35
- 3.2. Notations . . . 36
- 3.3. Method . . . 37
  - 3.3.1. Semi-dense image alignment . . . 38
  - 3.3.2. Plane-sweeping stereo . . . 41

3.3.3. Temporal motion stereo 43
3.4. Experimental evaluation 44
3.5. Conclusion 49

4. Robust VO with a Multi-Camera System 51
4.1. Introduction 51
4.2. Notations 52
4.3. Method 53
4.3.1. Tracker 54
4.3.2. Keyframe and feature selections 56
4.3.3. Local mapper 57
4.4. Experimental evaluation 61
4.5. Conclusion 67

II. Deep CNN Enhanced Images 69

5. Self-supervised Motion Deblurring 71
5.1. Introduction 71
5.2. Method 73
5.2.1. Deblurring and optical flow 74
5.2.2. Reblurring 75
5.2.3. Image warping 78
5.2.4. Relationship between $\mathbf{u}_{a\to b}/\mathbf{u}_{b\to a}$ and $\mathbf{u}$ 79
5.2.5. Loss functions 79
5.2.6. Occlusion handling 80
5.2.7. Differences with the method proposed by Chen et al. 80
5.3. Experimental evaluation 81
5.4. Conclusion 87

6. Deep Shutter Unrolling Network 101
6.1. Introduction 101
6.2. Method 104
6.2.1. Rolling shutter image formation model 104
6.2.2. Rolling shutter effect removal 105

6.2.3. Differentiable forward warping block 106
6.2.4. Network architecture 109
6.2.5. Loss functions 113
6.3. Datasets 113
6.4. Experimental evaluation 115
6.5. Conclusion 121

III. Algorithmic Perspective 125

7. Motion Blur Aware Robust Visual Odometry 127
7.1. Introduction 127
7.2. Preliminaries 129
7.3. Method 130
7.3.1. Motion blur image formation model 131
7.3.2. Direct image alignment with sharp images 131
7.3.3. Motion trajectory modeling 133
7.3.4. Direct image alignment with blurry images 136
7.3.5. More details on the transfer 137
7.4. Datasets 142
7.5. Experimental evaluation 144
7.6. Conclusion 153

8. Is Single Image Motion Estimation Possible? 155
8.1. Introduction 155
8.2. Single image motion estimation network 157
8.3. Upper bound motion cues that a single image motion estimation network can exploit 162
8.4. How is the performance of existing networks on dataset with complex motions? 164
8.5. Clarification 169
8.6. Conclusion 169

9. Conclusion and Outlook 171
9.1. Conclusion 171

9.2. Future works 173

Bibliography **175**

1. Introduction

Visual odometry (VO) is a technique used to determine the position and orientation of a camera (or a multi-camera rig) by using acquired images. As a fundamental block for many applications (e.g. robotics, augmented/virtual/mixed reality), VO has achieved great progress in both robotic and computer vision communities over the last decades. Many methods have been proposed to improve its accuracy, efficiency and robustness, which range from classical geometric approaches, deep learning based approaches to hybrid approaches.

Classical geometric approaches recover the camera motion from multi-view constraints. These methods can be further divided into direct approaches and sparse feature-based approaches. Direct approach relies on the photometric consistency assumption across multiple views within a short time interval. They jointly optimize the camera poses, 3D scene structure as well as camera intrinsic parameters by maximizing the photometric consistency. The representative works are LSD-SLAM [ESC14], DSO [EKC17] and their many variants [SDU+18, SDvS+19, LHSP17, LGH+18, MvSU+18, GWDC18]. Different from direct method, sparse feature-based methods extract a sparse set of keypoints from the raw images which are then matched across different views. Both the camera poses and 3D scene geometry are estimated by enforcing consistency between the keypoint locations and the projections of the scene structure.

End-to-end deep learning-based approaches usually formulate the problem as an end-to-end regression problem. The inputs to the network are usually a short sequence of consecutive frames. The network then predicts the relative poses between a target frame and other frames. The networks are trained either from ground truth motions (e.g. can be obtained from accurate real-time kinematic positioning system or other approaches) [UZU+17, XWL+19] or via self-supervision [ZBSL17, YS18].

Although many state-of-the-art algorithms have been proposed in the literature, their performance is still infancy compared to classical geometric based approaches. It is challenging for current state-of-the-art methods to achieve competitive performance for large-scale environments as classical approaches.

Hybrid methods try to embed deep networks into classical geometric frameworks [TTLN17a, YWSC18a, YSWC20, BCC+18b, ZBLD19]. These frameworks aim to leverage the benefits of both approaches to improve the performance of visual odometry. For example, both Tateno et al. [TTLN17a] and Yang et al. [YWSC18a] leverage the learned priors on the metric scale of surrounding objects to address the metric scale ambiguity issue of a classical monocular visual odometry algorithm. Bloesch et al. [BCC+18b] and Zhi et al. [ZBLD19] try to improve the estimated dense depth map or semantic labels by taking advantages of the priors learned by a deep neural network. Those algorithms still adopt the classical pipeline as the main backbone and then augment/replace part of the modules by a deep network. They usually can achieve state-of-the-art performance compared to their classical counterpart.

1.1. A classical VO pipeline

Since our work mainly focuses on classical geometric based VO algorithms, let us review the common blocks for a classical VO pipeline. Classical VO algorithms typically can be categorized to filtering based approach and batch optimization based approach. Batch optimization based approach usually can achieve more accurate motion estimations, while is also more computational intensive. State-of-the-art methods (e.g. ORB-SLAM [MAT17a] and DSO [EKC17]) usually adopt the batch optimization approach nowadays.

Fig. 1.1 demonstrates the typical architecture of batch optimization based approaches. The pipeline usually consists of two main parts, i.e. a front-end motion tracker and a back-end mapper. The front-end usually estimates the camera pose of the latest frame in real time. To reduce pose drifts, keyframe mechanism is ususally adopted [KM07]. The back-end then builds new 3D

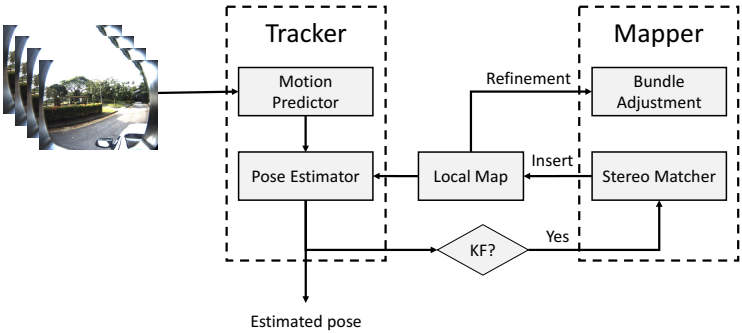


Figure 1.1.: **Overview of a classical VO pipeline.** The pipeline usually consists of two parts, i.e. a front-end motion tracker and a back-end local mapper. The tracker estimates the camera pose of latest frame in real-time. The local mapper jointly optimizes the camera poses and scene structures of recent keyframes, such that the pose drifts can be minimized.

maps for unexplored scenes and optimizes both the camera poses and scene structure jointly. Due to the high computational complexity of the back-end, it usually runs at a much lower frame rate (e.g. $2 \sim 3$ Hz) compared to the front-end.

The tracker usually consists of a motion predictor and a pose estimator. Constant velocity model is normally used to predict the camera pose of the latest frame, such that the pose estimator can have a good initial solution for least-square optimization. The pose estimator estimates the latest camera pose based on the information stored in the local map. Sparse feature based approach (e.g. ORB-SLAM [MAT17a]) usually extracts salient feature points from the latest frame, and then builds up correspondences with 3D landmarks stored in the local map for pose estimation. Different from sparse feature based approach, direct method (e.g. DSO [EKC17]) estimates the camera pose of current frame with respect to the latest keyframe via direct image alignment [BM04]. Forster et al. [FPS14] also propose a hybrid approach, which removes the feature extraction and matching step

in current frame. Instead, they track the sparse keypoints against current frame by appearance matching (i.e. direct approach). Once the positions of sparse keypoints are tracked successfully in the current frame, they build up 2D-3D correspondences with the 3D landmarks from the local map. Conventional re-projection errors are then used to estimate the camera pose. The experimental results demonstrate that it can effectively reduce the computational cost required by feature extraction and feature matching.

The back-end mapper typically consists of a stereo matcher and a sliding window optimizer. The stereo matcher is used to populate new 3D landmarks for un-explored areas. Depending on the hardware set-up, either temporal motion stereo or static stereo are used to estimate the 3D positions. To reduce pose drifts for long-term operation, the sliding window optimizer is usually used to refine both camera poses and scene structures jointly. For efficiency, the sliding window usually covers several latest keyframes. The information contained within the older keyframes can be retained by the state marginalization method [EKC17] or approach by graph optimization [MAT17a].

1.2. Contributions of the dissertation

While many state-of-the-art algorithms have been proposed, robustness is still a problem for visual odometry methods. To make VO useful for real-world applications, it should work robustly under various conditions. For example, to serve as a fundamental block for autonomous driving, robust vehicle motion estimation is critical for path/trajectory tracking, and environment perception. Motion estimation errors and failures might cause tragic accidents and limit wide-scale deployment of VO algorithms.

Several approaches have been proposed to improve the robustness of VO for specific environments. Alismail et al. [ABL16b] propose a dense binary descriptor that can be integrated within a multi-channel Lucas Kanade framework to improve illumination change robustness. Park et al. [PSP17] perform a systematic evaluation of real-time capable methods for illumination change robustness in direct visual SLAM. Zhang et al. [ZCS17] propose an active exposure control method for robust visual odometry in

high dynamic range (HDR) environments. For each frame, they choose an exposure time that maximizes an image quality metric. In the work from Pascoe et al. [PMT+17], a direct monocular SLAM algorithm based on the Normalized Information Distance (NID) metric is proposed. They show that the information-theoretic NID metric provides robustness to appearance variations due to lighting, weather, and structural changes in the scene. The early works from Pretto et al. [PMB+09] and Lee et al. [LKL11] have been proposed to improve the robustness of sparse keypoint based VO against motion blur. Pretto et al. [PMB+09] propose to detect motion blur robust sparse invariant features. The work from Lee et al. [LKL11] assumes the motion between neighbouring frames is smooth and try to linearly interpolate the motion within the exposure time. For each frame the initial motion is extrapolated from previous frames using a motion model and this prediction is used to re-blur the patches from the keyframe. The re-blurred patches are used to establish explicit sparse correspondences between the new frame and the keyframe. The camera poses and scene structure are then estimated from these correspondences.

In this dissertation, we propose methods which make VO robust from different perspectives, such that it can be better deployed to various real world scenarios. In particular, we improve the robustness of VO methods from the hardware perspective, the perspective of deep network enhanced input images, and the algorithmic perspective.

Hardware perspective: Large field of view is usually beneficial for environments with poor textures or dynamic environments. For example, in low-lighting environments (e.g. autonomous driving at night), the captured images usually have low contrast. It challenges both the feature based methods and direct methods, which usually rely on large image contrast (e.g. edge or corner features) for reliable motion estimation. A large field of view can thus increase the number of reliable features for a VO pipeline. Under this perspective, we explore the usuages of fisheye cameras and a multi-camera rig to improve the robustness of VO algorithms. Experimental results demonstrate that VO algorithm with a rig of multiple fisheye cameras can still estimate the camera motion reliably in dark environments at night; in contrast, a single stereo configuration is prone to failure due to the lack

of good texture.

Deep CNN enhanced input images perspective: Almost all VO algorithms assume the input images are of good quality. However, due to the environmental conditions, low image quality is sometimes unavoidable in real world applications, which can then drastically reduce the performance of VO systems. For example, one of the most common challenging cases, motion blurred images occur often in low-light environments where longer exposure times are necessary. This affects both feature based approaches (e.g. ORB-SLAM [MAT17a], SVO [FPS14]), which struggle to detect keypoints, and direct method (e.g. DSO [EKC17]) which rely on strong image gradients for their alignment. While relocalization strategies can partially mitigate the problem by allowing the VO to recover after losing track, there are many applications where it is critical to get camera pose estimates for each frame. For example to provide smoother user experiences in augmented reality. Another common challenging case is the image distortion due to rolling shutter effect. In contrast to a global shutter camera, which captures all pixels at the same time, a rolling shutter camera sequentially captures the image pixels row by row. Therefore, different types of distortions, e.g. skew, smear or wobble, will appear if the camera is moving during the image capture. Since most of the VO algorithms assume the input images are captured by global shutter cameras, rolling shutter distorted images would thus affect the performance of VO algorithms. Under this perspective, we propose to improve the quality of the captured images (i.e. image deblurring and rolling shutter effect correction) by deep convolutional neural networks (CNN). All the input images can thus be preprocessed before they are fed into a normal VO pipeline.

Algorithmic perspective: The above decoupled approaches for motion blur aware VO algorithm have several drawbacks: 1) Although recent deep network achieves remarkable performance on image deblurring, they still struggle to recover high quality sharp images from severely motion blurred input. In experiments we show that these recovery artifacts can still degrade the performance of visual odometry and lead to tracking failures. 2) State-of-the-art image deblurring networks require significant computation effort and usually cannot run in real time even with a high-end GPU [NKL17,

TGS+18, KBM+18]. This excludes their use for most visual odometry application scenarios. Recently Kupyn et al. [KMWW19] proposed a more efficient network for deblurring. However, we experimentally found that it has limited generalization performance for VO applications due to its reduced model capacity. Under this perspective, we propose a hybrid novel visual odometry method with direct approach that explicitly models and estimates the camera's local trajectory within exposure time. This allows us to actively compensate for any motion blur that occurs due to the camera motion, in a coupled way.

As the recent advances of deep learning technique in many computer vision tasks, several pioneering networks have been proposed to tackle the VO problem, with the aim to explore a new research direction to further improve the performance (e.g. in terms of accuracy and robustness) of VO algorithms. In this dissertation, we also try to study the limitations of existing learning based methods, such that it can provide beneficial insights for future research. By demonstrating the fact that a single image motion estimation network is sufficient to predict reasonable camera motions on KITTI dataset [GLU12], we conclude that KITTI odometry dataset [GLU12] might not be sufficient for those networks to evaluate on (those learning based pipelines are usually evaluated on KITTI dataset [GLU12] only). The reason is that the camera motions from KITTI dataset [GLU12] are usually constrained by the street directions, from which a single image already embeds many motion cues for the network to exploit.

1.3. Related work

Before we dive into the details of our methods, let us have a brief review on the related work. We present the related work from three different research areas, i.e. classical geometric based visual odometry, motion deblurring and rolling shutter effect removal, which are the most related to our dissertation.

1.3.1. Visual odometry

We define four different taxonomies, i.e., number of used cameras, inference techniques, feature representations, and methods with additional sensors. Furthermore, we would like to focus on conventional RGB cameras only (instead of depth cameras and event cameras). VO is an integral part of visual Simultaneous Localization and Mapping (SLAM). When we refer to VO in the context of a SLAM algorithm, we refer to the algorithm ignoring the loop-closure and re-localization parts. Since there are numerous papers on this problem, we focus on a subset of representative state-of-the-art algorithms instead of listing all related techniques. A more detailed review on VO and SLAM techniques can be found in [SF11, FS12, CCC+16].

Number of used cameras: The two most commonly used configurations are the monocular and the stereo settings. Approaches which use multiple cameras to build up a generalized camera system have also been proposed in the literature. We will describe their details in the following.

To the best of our knowledge, the first work in real-time monocular camera SLAM can be attributed to Davison et al. [Dav03, DRMS07]. They propose a filtering-based algorithm which runs at 30 Hz using a single CPU thread on standard PC hardware. In contrast to [Dav03, DRMS07], Nister et al. [NNB04] propose a pure visual odometry algorithm using an iterative optimization approach. The proposed algorithm runs at video rates on a single thread. Later, a system to separate localization and mapping on two parallel threads is proposed by Klein et al. [KM07], which becomes a standard paradigm for future VO algorithms. This separation enables the system to produce high-quality maps by using more advanced but computational expensive inference techniques, while simultaneously achieving real-time localization and mapping. The proposed system is mainly designed for a small AR workspace. Muartal et al. [MAMT15] further improve the system to handle large scale environments by incorporating re-localization and loop closure modules into the system. Concurrently, Engel et al. [ESC14] propose another novel monocular SLAM systems. They also follow the two-thread paradigm, but with new feature representations and inference techniques in contrast to the aforementioned algorithms. In particular, instead of using conventional sparse corner features, Engel

et al. [ESC14] uses all high gradient pixels for pose estimation directly. Camera pose is tracked by minimizing the photometric intensity errors. Engel et al. [EKC17] also proposed a new monocular camera VO system, which shares the same front-end as [ESC14] but has a more advanced and novel back-end for both pose and structure estimation.

One of the main drawbacks of monocular camera VO is its inability to recover metric scale. Therefore, to make them more useful for robotic applications, additional sensor modalities [ZS15] or cameras [LFP11, CLFP10] are typically employed to capture scale information. For most of the above mentioned monocular systems, there exists a stereo version which allows for metric scale recovery. For example, [MAT17a] extends ORB-SLAM from [MAMT15] to stereo camera configurations. Both [ESC14] and [FPS14] have also been extended to the stereo setting as proposed in [ESC15, FZG+17, LHSP17]. For most robotic applications, a wide field of view (FoV) or even 360° FoV is necessary for better situational awareness and perception abilities. Therefore, a generalized camera system comprising more than two cameras is favorable. For example, [HLP15, HLP14] successfully demonstrated a multi-camera SLAM system for a micro aerial vehicle (MAV). [FSR+13, HLP13] employed a multi-camera system, made up by four fisheye cameras to provide 360° FoV for a ground vehicle.

Inference techniques: There are two main types of techniques to infer latent camera poses and 3D structures given observed images. The first is the filtering-based technique. A representative for filtering-based technique is the MonoSLAM system from Davison et al. [Dav03]. In this system, an extended Kalman filter (EKF) is utilized for simultaneous localization and mapping. The camera poses, velocities and 3D feature positions are combined to form a high-dimensional state vector. As for most filtering-based techniques, this inference technique contains two steps, i.e. a prediction step and an update (correction) step. Prediction is performed by the underlying dynamic motion model and is used for the blind interval when no image observations are available. The update step is used to correct accumulated errors (and minimize corresponding variances) from the prediction step using new observations. In contrast to filtering-based approaches, which usually track only a small subset of camera poses and 3D structure, batch

optimization-based methods (bundle adjustment) iteratively refine the full structure and the poses of all keyframes. Therefore, Klein et al. [KM07] advance the technique by separating localization and mapping into two threads, so that they can use a computational intensive but more accurate bundle adjustment approach to refine the estimated camera poses and 3D structures. While the localization thread runs at frame rate, the mapping thread runs at a much lower frequency.

In the following, we provide an overview over batch optimization based approaches for state estimation. There are two variants of the batch optimization based approach. Depending on the used feature representations, which will be described in next section, they can be classified into two categories, i.e. geometry based approaches and photometric based approaches. Geometry based approaches refine either camera poses or 3D structures by minimizing feature re-projection errors. Representative systems include PTAM [KM07] and ORB-SLAM [MAMT15]. Photometric-based approaches minimize the pixel intensity errors directly to estimate either camera poses or 3D geometry. Representative systems include LSD-SLAM [ESC14] and DSO [EKC17]. Similar to [EKC17], Alismail et al. [ABL16a] also propose a photometric bundle adjustment algorithm for visual SLAM. Forster et al. [FPS14] also propose a hybrid approach, which combines both photometric based approach and geomtry based approach.

Both variants possess advantages and disadvantages. Geometry-based approaches are usually more robust to illumination changes which photometric approaches suffer from. Furthermore, geometry-based approaches are also more robust to dynamic scenes due to the usage of robust algorithms (e.g., RANSAC) for feature matching. However, RANSAC is usually relatively time-consuming. In contrast, photometric-based approaches directly use the image pixel intensities as features and avoid the feature extraction and matching steps. Recently, the newly proposed system by Engel et al. [EKC17] demonstrates that photometric-based approaches are able to outperform geometry-based approaches in terms of accuracy and efficiency on challenging datasets.

Feature representations: As discussed in the previous paragraph, there are two main types of feature representations used in VO algorithms: sparse

feature-based representations and direct pixel-based representations, typically considered at locations where the image gradient is high.

Sparse feature-based VO typically exploits corners as interest points. The feature descriptor is then computed from the image content in a local neighborhood around each detected corner. There exist a large variety of feature descriptors in the literature, e.g., FAST [RD06], SIFT [Low04], SURF [BTG06], ORB [RRKB11] et al. Most feature descriptors are designed for robustness against illumination changes, scale changes and other geometry changes. While sparse feature-based VO (e.g., ORB-SLAM) leads to a relatively easy optimization problem, those algorithms suffer from poorly textured scenes due to the small number of corner features (e.g. in indoor environment which usually contains more edge features) which can be used to establish sufficienct number of correspondences.

To overcome these problems, [ESC14, TNPH15, EKC17] propose to use pixels with high gradients (semi-dense) directly as features. The advantage of such an approach is that it can take advantage of image edges in addition to classical corner features, which helps in particular in the presence of poorly textured scenes. However, a good pose initialization is important for direct methods as they are prone to local minima during the optimization. Furthermore, the depth of each image pixel with high gradients need to be estimated simultaneously with the camera poses. Experimental results demonstrate that direct methods can be implemented efficiently and are often more robust to poor textured scenes as well as motion blur degraded images compared to conventional sparse feature-based methods. A hybrid approach from [FPS14] extracts FAST features and uses the photometric-based approach for efficient camera pose tracking, while reverts to geometry-based bundle adjustment for back-end. The proposed system is extremely efficient and runs at 300 Hz on a consumer laptop. The system from Engel et al. [EKC17] demonstrates that even an approach which considers all high gradient pixels can run in real-time.

Inertial aided VO: As mentioned in the previous sections, one of the main drawbacks for monocular VO is its inability to recover metric scale. Thus, additional sensor modalities or cameras are usually used for this purpose. For example, [ZS15] integrate Lidar and visual measurements

together to obtain a fast and robust odometry system, which currently ranks at top position in the KITTI VO benchmark [GLU12]. The benefits of using inertial measurements have also been successfully demonstrated in several state-of-the-art visual inertial odometry pipelines [MR07b,LLB+15, FCDS17]. In this section, we review inertial sensor-aided VO pipelines.

One of the pioneering state-of-the-art VIO systems can be attributed to MSCKF from [MR07b]. They tightly couple visual measurements and inertial measurements using an EKF. The inertial measurements are used to drive the rigid-body kinematic model-based prediction step, while the visual measurements are used for the update (correction) step. The resulting system is efficient and achieves good performances in terms of accuracy. Leutenegger et al. [LLB+15] later advance the technique by proposing a new algorithm based on batch optimization (i.e. *aka.* OKVIS). One of the main drawbacks of OKVIS is that the IMU measurements must be re-integrated in every optimization iteration which is time-consuming. Therefore, Forster et al. [FCDS17] propose an algorithm to address this problem. They pre-integrate the IMU measurements before the optimization starts. For each optimization step, they use these pre-integrated "measurements" only, saving computational resources. Experimental results show that the system outperforms both OKVIS and MSCKF in terms of accuracy and efficiency. The system is able to run at more than 50 Hz on a standard laptop.

Besides the ability to observe metric scale, inertial measurements also make the VO algorithm more robust due to their complementary nature wrt. the image observations. Inertial measurements are usually sampled at much higher frequencies ($>$200 Hz) compared to visual measurements ($\sim$30 Hz). They can capture the camera motion and dynamics better, which is important for robust visual tracking. Furthermore, visual tracking is susceptible to fast motion as features are easily lost and direct methods can get stuck in a local optimum. Inertial measurements can help to overcome outliers or sensor occlusions. Therefore, integrating inertial measurements is also beneficial for stereo-based visual odometry systems [UESC16] or multi-camera approaches [HLP15, FSR+13], where metric scale is directly available even without inertial measurements.

1.3.2. Image motion deblurring

Motion deblurring methods can be categorized into two groups: those that assume spatially uniform blur [Ric72, KF09, XJ10, FSH+06, SJA08, LWDF09, CL09] and those considering spatially varying blur [WSZP10, GJZ+10, TTB11]. Uniform deblurring methods assume that the blur kernel is identical for each pixel of the input image. Spatially varying deblurring methods assume that the blur kernels for each pixel may change with respect to its spatial location. Motion deblurring methods can also be classified into non-blind deblurring [Ric72, KF09, TTB11] and blind deblurring [XJ10, FSH+06, SJA08, LWDF09, CL09, WSZP10, GJZ+10] methods. Non-blind deblurring methods assume a known blur kernel to recover the latent sharp image. In contrast, blind deblurring methods need to simultaneously recover both the latent image and the blur kernel. Existing techniques can also be classified into two categories, i.e. optimization-based methods and deep learning based methods. We will detail them as follows.

Optimization-based methods: Optimization-based methods rely on the motion blur formation model to recover the latent sharp image by minimizing an energy function [KF09, XJ10, FSH+06, SJA08, LWDF09, CL09, ZXJ13], e.g., using Gaussian [KF09, XJ10, CL09] or Poisson [Ric72, TTB11] likelihood functions in the context of maximum-a-posteriori (MAP) estimation. Depending on the number of input blurry images, additional terms can be formulated by warping the other image(s) to a reference image using either dense flow or by combining relative camera poses with dense depth maps [KL15, PL17]. Due to the nonlinear and ill-posed (in the case of a single image) nature of the problem, prior information on either the motion blur kernel or the latent sharp image must be used to constrain the solution space [KF09, XJ10, FSH+06, SJA08, KF09, XJ10, CL09]. While optimization based approaches often offers better generalization performance, they are usually computational expensive, which prevents them from time constrained applications.

Deep Learning based methods: Deep learning based methods use convolutional neural networks (CNNs) to recover the latent sharp image. Xu et al. [XRLJ14] propose a CNN with two sub-networks, a two-hidden-layer

deconvolution network and a two-hidden-layer outlier rejection network. The network is trained end-to-end with known ground truth sharp images. An even deeper network with 15 layers was proposed in [HKZ15] for text image deblurring. [NKL17] further increased the number of layers to 40 in a multi-scale manner, resulting in a network with 120 layers for three scales. To further improve the network performance, an adversarial loss [GPM+14] was used in [KBM+18]. [TGS+18, ZPR+18] use recurrent neural networks for single image deblurring.

More recently, network architectures for multi-frame inputs, which are able to exploit temporal information, have been proposed [SDW17, WHS17, KLSH17]. Su et al. [SDW17] uses a network with skip connections for video deblurring and Wieschollek et al. [WHS17] exploit temporal information using a recurrent architecture. A spatio-temporal recurrent architecture with a small computational footprint was proposed in [KLSH17].

Deep learning-based approaches usually outperform optimization-based methods in terms of both efficiency and image quality. However, nearly all of the recent deep learning based methods are trained in a fully supervised manner with only few notable exceptions. Madam et al. [MKA18] adapted CycleGAN [ZPIE17] to the single image deblurring task. Using unpaired sharp images for training, they obtained good performance in the specific domain of images (e.g., text, faces). Chen et al. [CGG+18] propose to include a self-consistency loss for supervision. However, they report that their self-supervised model leads to degenerated solutions and hence optimize a hybrid loss function which heavily relies on supervision in the form of sharp images.

1.3.3. Rolling shutter effect removal

We categorize the related work on rolling shutter effect removal into classical approaches and deep learning based approaches. The classical approaches can be further classified into single image based and multiple image based approaches.

Classical single image based approaches: Rengarajan et al. [RRA16] propose to take advantage of the "straight lines must remain straight" as-

sumption to rectify a single rolling shutter image. The camera motion is assumed to be purely rotational. Curves are extracted and the motion is iteratively estimated by enforcing the transformed curves to be straight. Purkait et al. [PZL17] assumes the 3D scene captured by the camera obeys Manhattan world assumption [CY00]. The distortion is corrected by jointly aligning the vanishing directions. Lao and Ait-Aider [LAA18] propose a minimal solver to estimate the camera motion based on four straight lines from a single image. The motion is parameterized by pure rotations. The RANSAC algorithm is used to remove outliers such that the camera motion can be estimated robustly. The rolling shutter effects can then be removed given the estimated motion.

Classical multiple image based approaches: Liang et al. [LCC08] estimates per-pixel motion vector to rectify a rolling shutter image. Block matching is used to find correspondences between two consecutive frames, such that the motion can be estimated. Forssén and Ringaby [FR10] assumes the camera has either pure rotation or in-plane translational motion. The camera motion is estimated by minimizing the re-projection errors between sparse corresponding points. A KLT tracker [BM04] is used to establish the sparse correspondences. Karpenko et al. [KJBL11] extends the work of [FR10] by using inertial measurements. They model the camera motion by pure rotations. The rolling shutter effect is removed by solving an optimization problem, which jointly stabilizes the video and calibrates the gyroscope. Baker et al. [BBKS10] estimates the per-pixel motion vector from a video sequence to correct the rolling shutter distortion. The motion is estimated via a constant affine or translational distortion model, and is estimated in up to 30 row blocks. Grundmann et al. [GKCE12] relaxes the constraints that a calibrated camera is required for rolling shutter effect removal. They model the motion between two neighbouring frames as a mixture of homography matrices. The mixture of homographies is estimated by minimizing the re-projection errors of corresponding points. Zhuang et al. [ZCL17] proposes to solve a dense SfM problem given two consecutive rolling shutter images. They estimate both the camera motion and dense depth map from dense correspondences. A minimal solver is proposed to estimate the camera motion. Both the depth map and motion

are further estimated/refined by minimizing the re-projection errors. Vasu et al. [VMR18] propose to solve occlusion aware rolling shutter correction problem using multiple consecutive frames. They model the 3D geometry as a layer of planar scenes. The depth, camera motion, latent layer mask and latent layer intensities are jointly estimated. The global shutter image is recovered by the proposed image formation model given all above estimations.

Deep learning based approaches: Rengarajan et al. [RBR17] propose to estimate the camera motion from a single rolling shutter image by using a deep network. They assume a simple affine motion model. The global shutter image is then recovered given the estimated motion. They train the network with synthetic data, which is generated by using the proposed motion model. Zhuang et al. [ZTJ+19] extends [RBR17] for depth aware rolling shutter effect correction from a single image. Two independent networks are used to predict the dense depth map and camera motion respectively. The global shutter image is then recovered as a post-processing step, given the estimated dense depth map and camera motion.

1.4. Overview of the dissertation

Our dissertation consists of seven chapters. We will briefly describe their details as follows. Chapter 2 presents necessary background knowledge, such that the dissertation is self-contained. We will present the details on improving the robustness of VO methods from the hardware perspective in Chapter 3 and Chapter 4. In particular, Chapter 3 presents the details on how we extend a direct visual odometry method with a fisheye stereo camera. Chapter 4 demonstrates the details on a multi-camera VO algorithm. The algorithms which improve the quality of captured images are presented in both Chapter 5 and Chapter 6. In particular, we present a self-supervised motion deblurring algorithm in Chapter 5 and a rolling shutter effect removal algorithm in Chapter 6. Lastly, we will present a hybrid approach which improves the performance of VO method against motion blur in Chapter 7. In Chapter 8, we study the limitations of existing learning based methods, such that future research could benefit from. Chapter 9 concludes

the dissertation and discusses possible future works in the area of robust 3D vision.

2. Preliminaries

In this chapter, we will present the necessary preliminary knowledge so that the dissertation can be more self-contained. In particular, we will present two commonly used camera models, i.e. the pinhole camera model and the unified camera model for a fisheye camera. We will also discuss the exponential function and the logarithm function on the Lie group in details, since they are commonly used in the implementation of a visual odometry pipeline. The state marginalization technique which is used to reduce the number of estimated states for visual odometry, will also be discussed.

2.1. Camera models

We define two abstract functions $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ and $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ to represent the basic functions of a camera model, i.e. projection function and back-projection function respectively. The projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is used to project a 3D scene point into a 2D pixel point and vice verse for $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$. There are several camera models to model different camera devices. Here we focus on the two commonly used camera models, i.e. the pinhole camera model for normal cameras and the unified camera model [MR07a] for fisheye cameras.

Pinhole camera model: The projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ of a pinhole camera model can be represented as

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(\mathbf{X}) = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix}, \tag{2.1}$$

where $\mathbf{x} = \begin{bmatrix} u & v \end{bmatrix}^T$ is the pixel coordinate, f_x, f_y, c_x and c_y are the

intrinsice parameters of a pinhole camera model, $\mathbf{X} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the corresponding 3D point. The back-projection function $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ can be derived as

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \pi^{-1}(\mathbf{x}, \lambda) = \lambda \cdot \begin{bmatrix} \frac{1}{f_x} & 0 & -c_x \\ 0 & \frac{1}{f_y} & -c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{2.2}$$

where λ is the depth of the 3D point $\mathbf{X}$.

From Eq. (2.1), we can derive the Jacobian matrix for its projection function as follows, which will be used in later sections for gradient-based optimization:

$$\mathbf{J}_\pi = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{bmatrix}, \tag{2.3}$$

$$= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \frac{\partial(\frac{x}{z})}{\partial x} & \frac{\partial(\frac{x}{z})}{\partial y} & \frac{\partial(\frac{x}{z})}{\partial z} \\ \frac{\partial(\frac{y}{z})}{\partial x} & \frac{\partial(\frac{y}{z})}{\partial y} & \frac{\partial(\frac{y}{z})}{\partial z} \\ \frac{\partial(1)}{\partial x} & \frac{\partial(1)}{\partial y} & \frac{\partial(1)}{\partial z} \end{bmatrix}, \tag{2.4}$$

$$= \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{x \cdot f_x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{y \cdot f_y}{z^2} \end{bmatrix}. \tag{2.5}$$

Unified camera model: There are two steps required to project a 3D scene point $\mathbf{X}$ to a pixel point $\mathbf{x}$. First, $\mathbf{X}$ is projected onto a unit sphere as

$$\mathbf{X}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2+z^2}} \\ \frac{y}{\sqrt{x^2+y^2+z^2}} \\ \frac{z}{\sqrt{x^2+y^2+z^2}} \end{bmatrix}. \tag{2.6}$$

It is then projected to the pixel point $\mathbf{x}$ via

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \frac{x_s}{z_s+\xi} & \frac{y_s}{z_s+\xi} & 1 \end{bmatrix}^T, \tag{2.7}$$

where ξ is an intrinsic parameter of unified camera model and can be obtained by offline calibration.

Similarly, there are also two steps to compute $\mathbf{X}$ from $\mathbf{x}$. First, $\mathbf{x}$ is back-projected to the normalized plane (i.e. with $\lambda = 1$) as $\bar{\mathbf{X}}$ by Eq. (2.2). We can then lift $\bar{\mathbf{X}} = \begin{bmatrix} \bar{x} & \bar{y} & 1 \end{bmatrix}^T$ to the unit sphere by

$$\mathbf{X}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \begin{bmatrix} \alpha\bar{x} \\ \alpha\bar{y} \\ \alpha - \xi \end{bmatrix}, \tag{2.8}$$

where $\alpha = \frac{\xi + \sqrt{1+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+1}$. If we know the ray depth corresponding to $\mathbf{x}$ as d, then we can obtain $\mathbf{X}$ as

$$\mathbf{X} = d \cdot \mathbf{X}_s. \tag{2.9}$$

The Jacobian matrix of the projection function for a unified camera model can be derived from both Eq. (2.6) and Eq. (2.7). In particular, the Jacobian matrix for projecting 3D scene point from $\mathbf{X}$ to $\mathbf{X}_s$ can be derived as

$$\mathbf{J}_1 = \begin{bmatrix} \frac{\partial x_s}{\partial x} & \frac{\partial x_s}{\partial y} & \frac{\partial x_s}{\partial z} \\ \frac{\partial y_s}{\partial x} & \frac{\partial y_s}{\partial y} & \frac{\partial y_s}{\partial z} \\ \frac{\partial z_s}{\partial x} & \frac{\partial z_s}{\partial y} & \frac{\partial z_s}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{1}{a_2} - \frac{x^2}{a_3} & (-1) \cdot \frac{x \cdot y}{a_3} & (-1) \cdot \frac{x \cdot z}{a_3} \\ (-1) \cdot \frac{x \cdot y}{a_3} & \frac{1}{a_2} - \frac{y^2}{a_3} & (-1) \cdot \frac{y \cdot z}{a_3} \\ (-1) \cdot \frac{x \cdot z}{a_3} & (-1) \cdot \frac{y \cdot z}{a_3} & \frac{1}{a_2} - \frac{z^2}{a_3} \end{bmatrix} \tag{2.10}$$

where

$$a_1 = x^2 + y^2 + z^2, \tag{2.11}$$

$$a_2 = \sqrt{x^2 + y^2 + z^2}, \tag{2.12}$$

$$a_3 = (x^2 + y^2 + z^2)^{\frac{3}{2}}. \tag{2.13}$$

The Jacobian matrix for projecting $\mathbf{X}_s$ from unit sphere to pixel point $\mathbf{x}$

can be derived as

$$\mathbf{J}_2 = \begin{bmatrix} \frac{\partial u}{\partial x_s} & \frac{\partial u}{\partial y_s} & \frac{\partial u}{\partial z_s} \\ \frac{\partial v}{\partial x_s} & \frac{\partial v}{\partial y_s} & \frac{\partial v}{\partial z_s} \end{bmatrix} \tag{2.14}$$

$$= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \frac{\partial(\frac{x_s}{z_s+\xi})}{\partial x_s} & \frac{\partial(\frac{x_s}{z_s+\xi})}{\partial y_s} & \frac{\partial(\frac{x_s}{z_s+\xi})}{\partial z_s} \\ \frac{\partial(\frac{y_s}{z_s+\xi})}{\partial x_s} & \frac{\partial(\frac{y_s}{z_s+\xi})}{\partial y_s} & \frac{\partial(\frac{y_s}{z_s+\xi})}{\partial z_s} \\ \frac{\partial(1)}{\partial x_s} & \frac{\partial(1)}{\partial y_s} & \frac{\partial(1)}{\partial z_s} \end{bmatrix} \tag{2.15}$$

$$= \begin{bmatrix} \frac{f_x}{z_s+\xi} & 0 & -\frac{x_s \cdot f_x}{(z_s+\xi)^2} \\ 0 & \frac{f_y}{z_s+\xi} & -\frac{y_s \cdot f_y}{(z_s+\xi)^2}. \end{bmatrix} \tag{2.16}$$

Thus, the Jacobian matrix for the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ can be computed by the chain rule as

$$\mathbf{J}_\pi = \mathbf{J}_2 \cdot \mathbf{J}_1. \tag{2.17}$$

2.2. Exponential map

The exponential map operator connects the Lie algebra (e.g. $\mathbf{so}(3)$) to its Lie group (e.g. $\mathbf{SO}(3)$). We can formally define the exponential map as $\exp : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ for $\mathbf{SO}(3)$. The rotation matrix is parameterized by unit quaternion representation. If we further denote the tangent vector $\mathbf{r} = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}^T \in \mathbf{so}(3)$ and use the unit quaterion $\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$ to represent the rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$, we can have

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} = \exp(\mathbf{r}) = \begin{bmatrix} \lambda r_x \\ \lambda r_y \\ \lambda r_z \\ \cos(\theta) \end{bmatrix}, \tag{2.18}$$

where $\lambda = \frac{\sin(\theta)}{2\theta}$ and $\theta = \frac{1}{2}\sqrt{r_x^2 + r_y^2 + r_z^2}$. We therefore can derive the Jacobian $\frac{\partial \bar{\mathbf{q}}}{\partial \mathbf{r}} \in \mathbb{R}^{4\times3}$ as follows:

$$\frac{\partial \bar{\mathbf{q}}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial q_x}{\partial r_x} & \frac{\partial q_x}{\partial r_y} & \frac{\partial q_x}{\partial r_z} \\ \frac{\partial q_y}{\partial r_x} & \frac{\partial q_y}{\partial r_y} & \frac{\partial q_y}{\partial r_z} \\ \frac{\partial q_z}{\partial r_x} & \frac{\partial q_z}{\partial r_y} & \frac{\partial q_z}{\partial r_z} \\ \frac{\partial q_w}{\partial r_x} & \frac{\partial q_w}{\partial r_y} & \frac{\partial q_w}{\partial r_z} \end{bmatrix}, \tag{2.19}$$

where

$$\frac{\partial \theta}{\partial r_x} = \frac{r_x}{2\theta}, \qquad \frac{\partial \theta}{\partial r_y} = \frac{r_y}{2\theta}, \tag{2.20}$$

$$\frac{\partial \theta}{\partial r_z} = \frac{r_z}{2\theta}, \qquad \frac{\partial \lambda}{\partial \theta} = \frac{\cos(\theta) - 2\lambda}{2\theta}, \tag{2.21}$$

$$\frac{\partial q_x}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_x + \lambda, \qquad \frac{\partial q_x}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_x, \tag{2.22}$$

$$\frac{\partial q_x}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_x, \qquad \frac{\partial q_y}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_y, \tag{2.23}$$

$$\frac{\partial q_y}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_y + \lambda, \qquad \frac{\partial q_y}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_y, \tag{2.24}$$

$$\frac{\partial q_z}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_z, \qquad \frac{\partial q_z}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_z, \tag{2.25}$$

$$\frac{\partial q_z}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_z + \lambda, \qquad \frac{\partial q_w}{\partial r_x} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_x}, \tag{2.26}$$

$$\frac{\partial q_w}{\partial r_y} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_y}, \qquad \frac{\partial q_w}{\partial r_z} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_z}. \tag{2.27}$$

It can be seen that $\lambda = \frac{\sin(\theta)}{2\theta}$ is not well defined if $\theta \to 0$. It is special

handled by Taylor series expansion as follows

$$q_x = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_x, \tag{2.28}$$

$$q_y = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_y, \tag{2.29}$$

$$q_z = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_z, \tag{2.30}$$

$$q_w = 1 - \frac{1}{2}\theta^2 + \frac{1}{24}\theta^4. \tag{2.31}$$

Since $\theta \to 0$, the corresponding Jacobian is then derived as

$$\frac{\partial q_x}{\partial r_x} = 0.5, \quad \frac{\partial q_x}{\partial r_y} = 0, \quad \frac{\partial q_x}{\partial r_z} = 0, \tag{2.32}$$

$$\frac{\partial q_y}{\partial r_x} = 0, \quad \frac{\partial q_y}{\partial r_y} = 0.5, \quad \frac{\partial q_y}{\partial r_z} = 0, \tag{2.33}$$

$$\frac{\partial q_z}{\partial r_x} = 0, \quad \frac{\partial q_z}{\partial r_y} = 0, \quad \frac{\partial q_z}{\partial r_z} = 0.5, \tag{2.34}$$

$$\frac{\partial q_w}{\partial r_x} = 0, \quad \frac{\partial q_w}{\partial r_y} = 0, \quad \frac{\partial q_w}{\partial r_z} = 0. \tag{2.35}$$

2.3. Logarithm map

The logarithm map operator is the inverse of the exponential map operator. It can be formally defined as $\log : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ for rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$. We can have the tangent vector $\mathbf{r} \in \mathbf{so}(3)$ if we represent $\mathbf{R}$ with unit quaternion representation $\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$,

$$\mathbf{r} = \log(\bar{\mathbf{q}}) = \lambda \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}, \tag{2.36}$$

where $\lambda = 2\frac{\arctan(\frac{\theta}{q_w})}{\theta}$ and $\theta = \sqrt{q_x^2 + q_y^2 + q_z^2}$. We therefore can derive the Jacobian $\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}} \in \mathbb{R}^{3\times4}$ as follows:

$$\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}} = \begin{bmatrix} \frac{\partial r_x}{\partial q_x} & \frac{\partial r_x}{\partial q_y} & \frac{\partial r_x}{\partial q_z} & \frac{\partial r_x}{\partial q_w} \\ \frac{\partial r_y}{\partial q_x} & \frac{\partial r_y}{\partial q_y} & \frac{\partial r_y}{\partial q_z} & \frac{\partial r_y}{\partial q_w} \\ \frac{\partial r_z}{\partial q_x} & \frac{\partial r_z}{\partial q_y} & \frac{\partial r_z}{\partial q_z} & \frac{\partial r_z}{\partial q_w} \end{bmatrix}, \tag{2.37}$$

where

$$\frac{\partial r_x}{\partial q_x} = \frac{\partial \lambda}{\partial q_x} q_x + \lambda, \qquad \frac{\partial r_x}{\partial q_y} = \frac{\partial \lambda}{\partial q_y} q_x, \tag{2.38}$$

$$\frac{\partial r_x}{\partial q_z} = \frac{\partial \lambda}{\partial q_z} q_x, \qquad \frac{\partial r_x}{\partial q_w} = \frac{\partial \lambda}{\partial q_w} q_x, \tag{2.39}$$

$$\frac{\partial r_y}{\partial q_x} = \frac{\partial \lambda}{\partial q_x} q_y, \qquad \frac{\partial r_y}{\partial q_y} = \frac{\partial \lambda}{\partial q_y} q_y + \lambda, \tag{2.40}$$

$$\frac{\partial r_y}{\partial q_z} = \frac{\partial \lambda}{\partial q_z} q_y, \qquad \frac{\partial r_y}{\partial q_w} = \frac{\partial \lambda}{\partial q_w} q_y, \tag{2.41}$$

$$\frac{\partial r_z}{\partial q_x} = \frac{\partial \lambda}{\partial q_x} q_z, \qquad \frac{\partial r_z}{\partial q_y} = \frac{\partial \lambda}{\partial q_y} q_z, \tag{2.42}$$

$$\frac{\partial r_z}{\partial q_z} = \frac{\partial \lambda}{\partial q_z} q_z + \lambda, \qquad \frac{\partial r_z}{\partial q_w} = \frac{\partial \lambda}{\partial q_w} q_z, \tag{2.43}$$

$$\frac{\partial \lambda}{\partial q_x} = \frac{2q_w - \lambda}{\theta^2} q_x, \qquad \frac{\partial \lambda}{\partial q_y} = \frac{2q_w - \lambda}{\theta^2} q_y, \tag{2.44}$$

$$\frac{\partial \lambda}{\partial q_z} = \frac{2q_w - \lambda}{\theta^2} q_z, \qquad \frac{\partial \lambda}{\partial q_w} = -2. \tag{2.45}$$

It can be seen that $\lambda = 2\frac{\arctan(\frac{\theta}{q_w})}{\theta}$ is not well defined if either $\theta \to 0$ or $q_w \to 0$. They are thus special handled as follows:

- If $\theta \to 0$, we can approximate λ with $\lambda = \frac{2}{q_w} - \frac{2\theta^2}{3q_w^3}$. The correspondng Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = 2\frac{1}{q_w} - \frac{4q_x}{3q_w^3}, \qquad \frac{\partial \lambda}{\partial q_y} = 2\frac{1}{q_w} - \frac{4q_y}{3q_w^3}, \tag{2.46}$$

$$\frac{\partial \lambda}{\partial q_z} = 2\frac{1}{q_w} - \frac{4q_z}{3q_w^3}, \qquad \frac{\partial \lambda}{\partial q_w} = -2\frac{1}{q_w^2} + 2\frac{\theta^2}{q_w^4}. \tag{2.47}$$

- If $q_w \to 0$ and $q_w > 0$, we can have $\lambda = \frac{\pi}{\theta}$. The corresponding Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = -\frac{\lambda}{\theta^2} q_x, \qquad \frac{\partial \lambda}{\partial q_y} = -\frac{\lambda}{\theta^2} q_y, \tag{2.48}$$

$$\frac{\partial \lambda}{\partial q_z} = -\frac{\lambda}{\theta^2} q_z, \qquad \frac{\partial \lambda}{\partial q_w} = 0. \tag{2.49}$$

- If $q_w \to 0$ and $q_w < 0$, we can have $\lambda = -\frac{\pi}{\theta}$. The corresponding Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = \frac{\lambda}{\theta^2} q_x, \qquad \frac{\partial \lambda}{\partial q_y} = \frac{\lambda}{\theta^2} q_y, \tag{2.50}$$

$$\frac{\partial \lambda}{\partial q_z} = \frac{\lambda}{\theta^2} q_z, \qquad \frac{\partial \lambda}{\partial q_w} = 0. \tag{2.51}$$

2.4. State marginalization

For a full SLAM problem, the state inference algorithm is usually formulated to maximize the joint posterior probability (MAP)

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} \ \Pr(\mathcal{X}|\mathcal{Z}) = \underset{\mathcal{X}}{\operatorname{argmax}} \ \Pr(\mathcal{Z}|\mathcal{X}) \Pr(\mathcal{X}), \tag{2.52}$$

where $\mathcal{X}$ is the set of all states, $\mathcal{X}^*$ is its corresponding optimal estimate, $\mathcal{Z}$ is the set of all measurements, $\Pr(\mathcal{Z}|\mathcal{X})$ is the likelihood function and $\Pr(\mathcal{X})$ is the prior probability of states in $\mathcal{X}$. A prior is usually only provided for the initial state, i.e., $\mathbf{x}_0$, to fix the coordinate frame (i.e. gauge freedom) used for the above inference algorithm. Thus, the above equation can be further simplified as

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} \ \Pr(\mathbf{x}_0) \Pr(\mathcal{Z}|\mathcal{X}). \tag{2.53}$$

The measurement model for a typical SLAM problem is usually formulated as

$$\mathbf{z}_{ij} = \mathrm{f}(\mathbf{x}_i, \mathbf{x}_j) + \boldsymbol{\eta}, \tag{2.54}$$

where $\boldsymbol{\eta}$ is usually modelled as zero mean white Gaussian noise with covariance as $\boldsymbol{\Sigma}_{ij}$, $\mathbf{z}_{ij}$ is a measurement from $\mathcal{Z}$, both $\mathbf{x}_i$ and $\mathbf{x}_j$ are two states to be estimated from $\mathcal{X}$. Depends on the measurement type, $\mathbf{x}_i$ and $\mathbf{x}_j$ can be two neighbouring states or not. For example, if $\mathbf{z}_{ij}$ is from odometry measurement or IMU, $\mathbf{x}_i$ and $\mathbf{x}_j$ are usually neighbouring states. If $\mathbf{z}_{ij}$ is visual measurement, $\mathbf{x}_i$ and $\mathbf{x}_j$ can be either neighbouring states or not. To be general, let us denote the measurement function as f(·), which in reality may be different for different kind of measurements. Thus, we can re-formulate the above maximization problem to a minimization problem equivalently by taking the negative-log function on above cost function as

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \ \frac{1}{2} \left\| \hat{\mathbf{x}}_0 - \mathbf{x}_0 \right\|^2_{\boldsymbol{\Sigma}_0} + \frac{1}{2} \sum_i \sum_j \left\| \mathbf{z}_{ij} - \mathbf{f}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\boldsymbol{\Sigma}_{ij}}, \tag{2.55}$$

where $\hat{\mathbf{x}}_0$ is a prior, $\boldsymbol{\Sigma}_0$ is usually set to be infinitesimal to fix $\hat{\mathbf{x}}_0$ during optimization, $\|\mathbf{r}\|^2_{\boldsymbol{\Sigma}}$ is defined to be equal to $\mathbf{r}^T \boldsymbol{\Sigma}^{-1} \mathbf{r}$ and is thus a scalar. The above cost function is usually optimized by gradient methods, e.g., Gauss-Newton method.

It is trivial to observe that the number of states increases without upper bound with time for an incremental SLAM problem. Thus, the complexity of full MAP estimation increases with time, which makes real-time inference impossible. In this section, we introduce a technique that can be used to optimally remove old states such that the total number of states used for MAP inference is bounded. In particular, the technique enables an approximation with constant complexity to the full MAP problem. The theory derived here will be used for the back-end fixed-lag smoother in our implementation.

To the best of my knowledge, the idea to remove states optimally for a SLAM problem is first proposed by Sibley et al. in [SMS10]. Dong-Si et al. further analyse and solve its consistency problem in [DSM11]. Both Leutenegger et al. [LLB+15] and Engel et al. [EKC17] then apply the formulations to their own odometry problems. Here we follow the formulations from [DSM11] which gives clear derivations and analysis of the estimator. To make the formulation general, let us denote the i^{th} state vector as $\mathbf{x}_i$. At timestamp k, we assume there are N states in total within

time interval $[0,\ k]$. We can further divide this set of N states into three disjoint subsets $\mathcal{X}_m$, $\mathcal{X}_r$ and $\mathcal{X}_o$ as following. Set $\mathcal{X}_m$ is defined to contain all the states that we are going to marginalize out/remove, which usually are well estimated old states. Set $\mathcal{X}_r$ contains all the states that we are going to keep but are connected to a state from $\mathcal{X}_m$, i.e., there is a measurement $\mathbf{z}$ depends on both of them. Set $\mathcal{X}_o$ is defined to contain all the remaining states, i.e., the states we are going to keep and have no relationship with $\mathcal{X}_m$, but a measurement $\mathbf{z}$ can possibly depend on a state from $\mathcal{X}_r$ and a state from $\mathcal{X}_o$. For the ease of the algorithm derivations, we can define

$$\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \mathbf{f}(\mathbf{x}_i, \mathbf{x}_j), \tag{2.56}$$

where $\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j)$ is the residual between real measurements and predicted measurements from both states $\mathbf{x}_i$ and $\mathbf{x}_j$. Now the full SLAM problem can thus be decomposed into following form

$$\begin{aligned}
\{\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_o\}^* = & \underset{\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \frac{1}{2} \left\| \hat{\mathbf{x}}_0 - \mathbf{x}_0 \right\|^2_{\mathbf{\Sigma}_0} && (2.57)\\
& + \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_m} \sum_{\mathbf{x}_j \in \mathcal{X}_m \cup \mathcal{X}_r} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}} && (2.58)\\
& + \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_r \cup \mathcal{X}_o} \sum_{\mathbf{x}_j \in \mathcal{X}_r \cup \mathcal{X}_o} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}}, && (2.59)\\
= & \underset{\mathcal{X}_m, \mathcal{X}_r}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_m} \sum_{\mathbf{x}_j \in \mathcal{X}_m \cup \mathcal{X}_r} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}} && (2.60)\\
& + \underset{\mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_r \cup \mathcal{X}_o} \sum_{\mathbf{x}_j \in \mathcal{X}_r \cup \mathcal{X}_o} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}}, && (2.61)
\end{aligned}$$

where $\mathbf{r}$ should be a valid residual, i.e., has a real corresponding physical measurements, and we combine Eq. (2.57) into Eq. (2.58) for simplicity since it can be considered as a well estimated old state with linear measurement function. As the above cost function is minimized after certain

iterations, we can approximate it with its 2^{nd} order Taylor series as follows

$$\mathrm{E}(\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_o) = \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_m} \sum_{\mathbf{x}_j \in \mathcal{X}_m \cup \mathcal{X}_r} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}} \tag{2.62}$$

$$+ \underset{\mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_r \cup \mathcal{X}_o} \sum_{\mathbf{x}_j \in \mathcal{X}_r \cup \mathcal{X}_o} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}}, \tag{2.63}$$

$$\approx \sum_{\mathbf{x}_i \in \mathcal{X}_m} \sum_{\mathbf{x}_j \in \mathcal{X}_m \cup \mathcal{X}_r} \{ \mathrm{g}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) + \mathbf{J}^T_{g_{ij}} \begin{bmatrix} \hat{\mathbf{x}}_i - \mathbf{x}_i \\ \hat{\mathbf{x}}_j - \mathbf{x}_j \end{bmatrix} \tag{2.64}$$

$$+ \frac{1}{2} \begin{bmatrix} \hat{\mathbf{x}}_i - \mathbf{x}_i \\ \hat{\mathbf{x}}_j - \mathbf{x}_j \end{bmatrix}^T \mathbf{H}_{g_{ij}} \begin{bmatrix} \hat{\mathbf{x}}_i - \mathbf{x}_i \\ \hat{\mathbf{x}}_j - \mathbf{x}_j \end{bmatrix} \} \tag{2.65}$$

$$+ \underset{\mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{X}_r \cup \mathcal{X}_o} \sum_{\mathbf{x}_j \in \mathcal{X}_r \cup \mathcal{X}_o} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}}, \tag{2.66}$$

where $\mathbf{J}_g$ and $\mathbf{H}_g$ are the Jacobian and Hessian matrices of function $\mathrm{g}(\mathbf{x}_i, \mathbf{x}_j)$ evaluated at current estimates $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ and function $\mathrm{g}(\mathbf{x}_i, \mathbf{x}_j)$ is defined as

$$\mathrm{g}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \left\| \mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \right\|^2_{\mathbf{\Sigma}_{ij}}. \tag{2.67}$$

Now it is trivial to observe that we can rewrite Eq. (2.64) and Eq. (2.65) in matrix form as

$$\mathrm{E}(\mathcal{X}_m, \mathcal{X}_r) = \mathrm{G} + \mathbf{J}^T \begin{bmatrix} \hat{\mathbf{X}}_m - \mathbf{X}_m \\ \hat{\mathbf{X}}_r - \mathbf{X}_r \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \hat{\mathbf{X}}_m - \mathbf{X}_m \\ \hat{\mathbf{X}}_r - \mathbf{X}_r \end{bmatrix}^T \mathbf{H} \begin{bmatrix} \hat{\mathbf{X}}_m - \mathbf{X}_m \\ \hat{\mathbf{X}}_r - \mathbf{X}_r \end{bmatrix}, \tag{2.68}$$

where G is a scalar and equals to the sum of all functions $\mathrm{g}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$, $\mathbf{X}_m$ and $\mathbf{X}_r$ are vectors containing all states from $\mathcal{X}_m$ and $\mathcal{X}_r$ respectively, both $\hat{\mathbf{X}}_m$ and $\hat{\mathbf{X}}_r$ are the current estimates for $\mathbf{X}_m$ and $\mathbf{X}_r$ respectively, both $\mathbf{J}$ and $\mathbf{H}$ are stacked based on $\mathbf{J}_{g_{ij}}$ and $\mathbf{H}_{g_{ij}}$ according to the ordering of

$\mathbf{X}_m$ and $\mathbf{X}_r$ as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_m \\ \mathbf{J}_r \end{bmatrix}, \tag{2.69}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix}. \tag{2.70}$$

Since Gauss-Newton method is usually used for the above cost function optimization, both $\mathbf{J}$ and $\mathbf{H}$ can readily be obtained from the last optimization iteration. At timestamp $k+1$, we can get new measurements as well to create new set of states $\mathcal{X}_n$. To keep the number of total states bounded, we can optimally remove all states from $\mathcal{X}_m$ before we incorporate new set of states $\mathcal{X}_n$. From energy function minimization perspective, optimally removing states from $\mathcal{X}_m$ is equivalent to minimizing the energy function, i.e. Eq. (2.68), with respect to all states $\mathbf{X}_m$. We can easily get the closed-form solution for the optimal $\mathbf{X}_m^*$ due to the quadratic form of Eq. (2.68). In particular, $\mathbf{X}_m^*$ can be obtained by taking gradient operator on both side of Eq. (2.68) with respect to $\mathbf{X}_m$ and then solve for $\mathbf{X}_m^*$ by setting $\frac{\partial \mathrm{E}(\mathcal{X}_m, \mathcal{X}_r)}{\partial \mathbf{X}_m} = 0$. After certain algebric manipulations, we can get

$$\begin{bmatrix} \mathbf{J}_m \\ \mathbf{J}_r \end{bmatrix}^T \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} \hat{\mathbf{X}}_m - \mathbf{X}_m \\ \hat{\mathbf{X}}_r - \mathbf{X}_r \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \mathbf{0}. \tag{2.71}$$

The above equation can further be simplified to be

$$\begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{X}}_m - \mathbf{X}_m \\ \hat{\mathbf{X}}_r - \mathbf{X}_r \end{bmatrix} = - \begin{bmatrix} \mathbf{J}_m \\ \mathbf{J}_r \end{bmatrix}, \tag{2.72}$$

in which we use the property $\mathbf{H} = \mathbf{H}^T$. Now we can solve $\hat{\mathbf{X}}_m - \mathbf{X}_m$ as

$$\hat{\mathbf{X}}_m - \mathbf{X}_m = (\mathbf{H}_{mm} - \mathbf{H}_{mr}\mathbf{H}_{rr}^{-1}\mathbf{H}_{rm})^{-1}(-\mathbf{J}_m + \mathbf{H}_{mr}\mathbf{H}_{rr}^{-1}\mathbf{J}_r). \tag{2.73}$$

By back-substituting $\hat{\mathbf{X}}_m - \mathbf{X}_m$ to Eq. (2.68), we can get

$$\mathrm{E}(\mathcal{X}_r) = \mathrm{E}(\mathcal{X}_m^*, \mathcal{X}_r) = \lambda + \mathbf{J}_{E_r}^T(\hat{\mathbf{X}}_r - \mathbf{X}_r) + \frac{1}{2}(\hat{\mathbf{X}}_r - \mathbf{X}_r)^T \mathbf{H}_{E_r}(\hat{\mathbf{X}}_r - \mathbf{X}_r), \tag{2.74}$$

where we have

$$\mathbf{J}_{E_r} = -\mathbf{J}_r + \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{J}_m, \tag{2.75}$$

$$\mathbf{H}_{E_r} = \mathbf{H}_{rr} - \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{H}_{mr}, \tag{2.76}$$

and they are all evaluated at current estimates of $\mathbf{X}_m$ and $\mathbf{X}_r$, i.e. at timestamp k. Now it is trivial to find that the full MAP estimator at timestamp k is permanently approximated by $\mathcal{X}_m^*$ with the benefit of using less states for the optimization. At timestamp $k+1$, we have new states $\mathcal{X}_n$, we can then approximate the full MAP estimator as follows.

$$\{\mathcal{X}_r^*, \mathcal{X}_o^*, \mathcal{X}_n^*\} = \underset{\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n) \tag{2.77}$$

$$= \underset{\mathcal{X}_m, \mathcal{X}_r}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_m, \mathcal{X}_r) + \underset{\mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_r, \mathcal{X}_o) \tag{2.78}$$

$$+ \underset{\mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n), \tag{2.79}$$

$$\approx \underset{\mathcal{X}_r}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_r) + \underset{\mathcal{X}_r, \mathcal{X}_o}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_r, \mathcal{X}_o) \tag{2.80}$$

$$+ \underset{\mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n}{\operatorname{argmin}} \mathrm{E}(\mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n), \tag{2.81}$$

where we assume $\mathcal{X}_n$ has no connection with $\mathcal{X}_m$, $\mathrm{E}(\mathcal{X}_r, \mathcal{X}_o, \mathcal{X}_n)$ contains only residuals involving at least one state from $\mathcal{X}_n$. The above energy function can then be optimized by Gauss-Newton method or other gradient method. It can be observed that only the prior term $\mathrm{E}(\mathcal{X}_r)$ affects the sparsity pattern of the Hessian matrix of above energy function. Dong-si et al. observes the problem that the Jacobian and Hessian matrices of $\mathrm{E}(\mathcal{X}_r)$ are to be evaluated at different value of $\mathbf{X}_r$ for optimization after timestamp k in [DSM11]. However, it can be observed that $\mathrm{E}(\mathcal{X}_r)$ is permenently approximated by its 2^{nd}-order Taylor expansion around the estimate of $\mathbf{X}_r$ at timestamp k. This inconsistent evaluation of the Jacobian and Hessian matrices will result in spurious measurements and make the estimator inconsistent. To solve this problem and to make the estimator more accurate, Dong-si et al. propose a simple solution in [DSM11]. In particular, they propose to fix the evaluation point of $\mathbf{X}_r$ for the Jacobian and Hessian

matrices at their estimates at timestamp k for future linearizations. We will adopt this solution in our future implementations.

Part I.

Hardware Perspective

3. Direct Visual Odometry for a Fisheye-Stereo Camera

3.1. Introduction

Forster et al. [FPS14] and Engel et al. [ESC14] started a new wave of semi-direct and direct visual odometry (VO) methods respectively for camera motion estimation. Different from previous sparse feature-based visual odometry methods [SF11, FS12], which usually require feature detection and matching for each frame, direct approaches estimate the camera poses and scene structures directly from raw pixel intensity values.

Semi-direct and direct visual odometry methods generally follow the simultaneous tracking and mapping paradigm [KM07]. These two types of methods are similar in the aspect that pixel intensity values are used for both motion estimation and stereo matching. The difference is that semi-direct methods optimize pose and structure by minimizing feature-based re-projection errors, while direct methods estimate pose and structure by minimizing photometric errors. Similar to sparse feature based VO methods, the semi-direct approach samples sparse corner features for motion estimation. However, direct approach usually samples a large number of pixels with high gradients (i.e. semi-dense). To simultaneously estimate the camera motion and do semi-dense 3D reconstruction, we adopt the direct approach for our algorithm.

The metric scale ambiguity is a problem for monocular VO algorithms [ESC14, FPS14]. The community has proposed the use of inertial measurement units (IMUs) [LLB+15, MR07b, TNPH15] or stereo cameras [CLFP10, LFP11] to recover the metric scale for real-world applications. We see a stereo camera as more robust than an IMU for automotive applica-

tions. The reason is that cars mostly accelerate during departure, breaking and turns. Most of the times, cars run at near constant speed, where the IMU has low signal to noise ratios and would be less reliable. To further improve the robustness of the proposed VO algorithm, we adopt fisheye cameras to enlarge the field of view, which has been shown is beneficial to the VO algorithm [ZRFS16]. It significantly increases the number of high gradient pixels, which enables the algorithm to estimate the camera motion more robust and accurate.

Therefore, we propose a direct visual odometry algorithm for a fisheye-stereo camera in this chapter, which is based on our publication [LHSP17]. Real world experimental results demonstrate that our algorithm not only gives low-drift motion estimation but also is able to do accurate semi-dense 3D reconstruction.

3.2. Notations

We denote the global world coordinate frame with $\mathcal{F}_W$, the stereo-camera coordinate frame with $\mathcal{F}_S$, and the individual camera coordinate frames with $\mathcal{F}_L$ and $\mathcal{F}_R$. We denote the image captured by the left camera at time step k as $\mathbf{I}_L^k$. Similarly, $\mathbf{I}_R^k$ denotes the image captured by the right camera at timestamp k. We denote the camera pose at time step k by a rigid body transformation matrix $\mathbf{T}_{S_k}^W \in \mathbf{SE}(3)$, which transforms points from the stereo coordinate frame $\mathcal{F}_S$ to the global world coordinate frame $\mathcal{F}_W$. The transformation matrix $\mathbf{T}_{S_k}^W$ is parameterized by a unitary quaternion and a translation vector. It is thus over-parameterized by 7 parameters, while it has only 6 degrees of freedom. Given the extrinsic parameters of the stereo camera, we are able to compute the poses of the left camera and the right camera as

$$\mathbf{T}_{L_k}^W = \mathbf{T}_{S_k}^W \cdot \mathbf{T}_L^S, \tag{3.1}$$

$$\mathbf{T}_{R_k}^W = \mathbf{T}_{S_k}^W \cdot \mathbf{T}_R^S, \tag{3.2}$$

where $\mathbf{T}_L^S$ and $\mathbf{T}_R^S$ are the extrinsic parameters of the left camera and the right camera respectively.

Given the fact that the rigid body transformation matrix is over parameterized, on-manifold optimization requires a minimal representation of the rigid body transformation. In this case, we use the Lie algebra $\mathbf{se}(3)$ corresponding to the tangent space of $\mathbf{SE}(3)$. We denote the algebra elements, also known as twist coordinates, with $\boldsymbol{\xi} = [\mathbf{v}, \boldsymbol{\omega}]^T$ where $\mathbf{v}$ is the linear velocity and $\boldsymbol{\omega}$ is the angular velocity. We use the exponential function to transform the twist coordinate $\boldsymbol{\xi} \in \mathbf{se}(3)$ to a rigid body transformation matrix $\mathbf{T} \in \mathbf{SE}(3)$:

$$\mathbf{T}(\boldsymbol{\xi}) = \exp(\boldsymbol{\xi}). \tag{3.3}$$

Similarly, we use the logarithm function to transform a rigid body transform matrix $\mathbf{T} \in \mathbf{SE}(3)$ to twist coordinate $\boldsymbol{\xi} \in \mathbf{se}(3)$:

$$\boldsymbol{\xi} = \log(\mathbf{T}). \tag{3.4}$$

The details on the exponential map function and the logarithm map function can be found from Chapter 2.

3.3. Method

In this section, we describe our semi-dense visual odometry algorithm for a fisheye-stereo camera. As shown in Fig. 3.1, our algorithm follows the simultaneous tracking and mapping paradigm [KM07], which encompasses a tracking thread and a mapping thread. The tracking thread tracks the current frame with respect to a keyframe. The mapping thread initializes and refines the depth map corresponding to the latest keyframe. Both static stereo and temporal stereo are used for depth initialization and refinement respectively.

The tracker consists of a model-based motion predictor, and a pose estimator. The motion predictor predicts the camera pose of current frame based on the constant velocity assumption. With the predicted pose, the pose estimator estimates the camera pose of current frame by direct image alignment. It maximizes the photometric consistency between the latest keyframe and current frame, which we will detail in the next section. For efficiency, we only track the frames captured by the left camera.

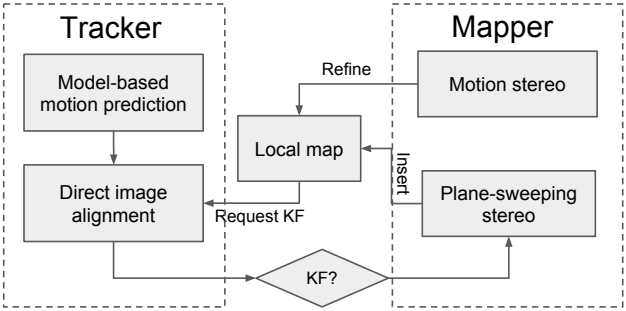


Figure 3.1.: System overview

To reduce pose drifts, we adopt the keyframe mechanism for the mapper. Current frame is selected as a keyframe if the relative pose with respect to its latest keyframe exceeding a pre-defined threshold. Since we are using fisheye cameras, conventional stereo matching algorithm, which assumes the input stereo images are pre-rectified would not work. We therefore use the plane-sweeping stereo matching method [HHL+15] to estimate the depth of newly sampled feature points from the latest created keyframe (i.e. static stereo). To further refine the semi-dense depth map, current frame (which is captured by the left camera) is also used for stereo matching (i.e. motion stereo). The relative pose between the current frame and the latest keyframe is provided by the tracker. We will detail them as follows.

3.3.1. Semi-dense image alignment

The tracking thread of our pipeline tracks the current frame against its latest keyframe by using semi-dense direct image alignment algorithm. Direct image alignment algorithm is the core of direct method. It estimates the camera pose by maximizing the photometric consistency across the frames. For our method, we estimate the camera pose of the current frame with

respect to its latest keyframe by minimizing following cost function:

$$\tilde{\mathbf{T}}_{ref}^{cur} = \underset{\mathbf{T}_{ref}^{cur}}{\operatorname{argmin}} \sum_{i} \left\| \mathbf{I}_{ref}(\mathbf{x}_i) - \mathbf{I}_{cur}(\hat{\mathbf{x}}_i) \right\|_2 , \tag{3.5}$$

where $\mathbf{T}_{ref}^{cur} \in \mathbf{SE}(3)$ is the transformation matrix from the reference keyframe to current frame, $\mathbf{I}_{ref}$ is the image of the reference keyframe, $\mathbf{I}_{cur}$ is the image of the current frame, $\mathbf{x}_i \in \mathbb{R}^2$ is one of the sampled high gradient pixels in the reference keyframe, and $\hat{\mathbf{x}}_i \in \mathbb{R}^2$ is the corresponding pixel of $\mathbf{x}_i$ in the current image. The camera pose of current frame in the global world coordinate can then be recovered by

$$\mathbf{T}_{cur}^{W} = \mathbf{T}_{ref}^{W} \cdot (\mathbf{T}_{ref}^{cur})^{-1}, \tag{3.6}$$

where $\mathbf{T}_{ref}^{W}$ is the camera pose of the latest keyframe in the global world coordinate frame. We can further have

$$\hat{\mathbf{x}}_i = \boldsymbol{\pi}(\mathbf{T}_{ref}^{cur} \cdot \boldsymbol{\pi}^{-1}(\mathbf{x}_i, d_i)), \tag{3.7}$$

where $\boldsymbol{\pi} : \mathbb{R}^3 \to \mathbb{R}^2$ is the camera projection function, which maps a 3D point to the 2D image plane; $\boldsymbol{\pi}^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the camera back-projection function, which transforms a 2D point with known depth d_i to a 3D point.

The cost function can be solved iteratively by using the Gauss-Newton method or the Levenberg-Marquardt method. To achieve better efficiency, we employ the inverse compositional algorithm [BM04]. It does one-time computation of the Jacobian and Hessian matrices instead of re-computing them for each iteration, so that the computational cost can be reduced. In particular, we minimize following cost function instead of Eq. (3.5) for each iteration:

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i} \left\| \mathbf{I}_{ref}(\pi(\mathbf{T}(\boldsymbol{\xi}) \cdot \mathbf{P}_i)) - \mathbf{I}_{cur}(\pi(\mathbf{T}_{ref}^{cur} \cdot \mathbf{P}_i)) \right\|_2 , \tag{3.8}$$

where $\boldsymbol{\xi} \in \mathbf{se}(3)$ is the twist coordinate on the manifold and we can convert it to a rigid body transformation matrix $\mathbf{T}(\boldsymbol{\xi}) \in \mathbf{SE}(3)$ via Eq. (3.3), $\mathbf{P}_i \in \mathbb{R}^3$ is the 3D coordinate cooresponding to pixel $\mathbf{x}_i \in \mathbb{R}^2$ with known

depth d_i, and it can be computed as $\mathbf{P}_i = \boldsymbol{\pi}^{-1}(\mathbf{x}_i, d_i)$. Using the chain rule, can we derive the Jacobian matrix:

$$\mathbf{J}_i = \nabla \mathbf{I}_{ref}|_{\mathbf{x}_i} \cdot \frac{\partial \boldsymbol{\pi}}{\partial \mathbf{P}_i} \cdot \frac{\partial (\mathbf{T}(\boldsymbol{\xi}) \cdot \mathbf{P}_i)}{\partial \boldsymbol{\xi}}|_{\boldsymbol{\xi} \to \mathbf{0}}, \tag{3.9}$$

where $\mathbf{J}_i \in \mathbb{R}^{1\times 6}$ is the partial Jacobian of the i^{th} residual induced by pixel $\mathbf{x}_i$, with respect to the twist coordinate $\boldsymbol{\xi}$ on the manifold; $\nabla \mathbf{I}_{ref}|_{\mathbf{x}_i} \in \mathbb{R}^{1\times 2}$ is the image gradients at pixel location $\mathbf{x}_i$; $\frac{\partial \boldsymbol{\pi}}{\partial \mathbf{P}_i} \in \mathbb{R}^{2\times 3}$ is the projection Jacobian at 3D position $\mathbf{P}_i$. According to the Gauss-Newton method, we can compute the optimal $\boldsymbol{\xi}$ with:

$$\boldsymbol{\xi} = -(\sum_i \mathbf{J}_i^T \mathbf{J}_i)^{-1} \cdot \sum_i r_i \mathbf{J}_i^T, \tag{3.10}$$

where $r_i \in \mathbb{R}$ is the residual which can be computed by

$$r_i = \mathbf{I}_{ref}(\mathbf{x}_i) - \mathbf{I}_{cur}(\pi(\mathbf{T}_{ref}^{cur} \cdot \mathbf{P}_i)). \tag{3.11}$$

The relative pose between the current frame and its reference keyframe can then updated iteratively by:

$$\mathbf{T}_{ref}^{cur} = \mathbf{T}_{ref}^{cur} \cdot \mathbf{T}(\boldsymbol{\xi})^{-1}. \tag{3.12}$$

We use the Huber robust function for the residual error r_i for each pixel to suppress the effect of outliers. To achieve better convergence performance, a coarse-to-fine approach by using multiple pyramid levels is also applied during the optimization. Furthermore, the initial relative pose $\mathbf{T}_{ref}^{cur} \in \mathbf{SE}(3)$ for the above optimization procedure is computed using a constant velocity motion prediction model as shown in Fig. 3.1.

It can be observed that the Jacobian matrix $\mathbf{J}_i$ will only need to be computed once during the optimization, since it does not rely on $\mathbf{T}_{ref}^{cur}$. For each optimization iteration, we will thus only need to compute the residuals r_i (i.e. Eq. (3.11)) for all the sampled high gradient pixels. Therefore, the least square optimization can be implemented efficiently.

3.3.2. Plane-sweeping stereo

For a pair of fisheye stereo images, pixel correspondences lie on epipolar curves instead of straight epipolar lines. Thus, conventional epipolar line disparity search algorithms cannot be used for fisheye stereo matching. To use fisheye images directly without rectifying them to pinhole images, we use the plane-sweeping stereo algorithm [HHL+15] for fisheye stereo matching to initialize the depth map of the keyframe. Plane-sweeping stereo assumes that scenes are locally planar and tests a set of plane hypotheses. These plane hypotheses are used to warp pixels from a reference view to the current view for similarity matching. The plane hypothesis which gives the maximum similarity measure is recorded and used to compute the ray depth of the corresponding pixel. The algorithm is implemented based on [HHL+15], and we will discuss the details as follows.

We define a set of plane hypotheses $\{\mathbf{n}_1, d_1\}, \ldots, \{\mathbf{n}_m, d_m\}, \ldots, \{\mathbf{n}_M, d_M\}$, where $\mathbf{n}_m \in \mathbb{R}^{3\times 1}$ and d_m are the normal and depth respectively of the m^{th} plane in the reference coordinate frame $\mathcal{F}_L$. For each sampled plane hypothesis, we compute its corresponding homography matrix $\mathbf{H}_L^R \in \mathbb{R}^{3\times 3}$ with:

$$\mathbf{H}_L^R = \mathbf{R}_L^R - \frac{\mathbf{t}_L^R \cdot \mathbf{n}_m{}^T}{d_m}, \tag{3.13}$$

where both $\mathbf{R}_L^R \in \mathbb{R}^{3\times 3}$ and $\mathbf{t}_L^R \in \mathbb{R}^{3\times 1}$ are the corresponding rotation matrix and translation vector of the extrinsic transformation matrix $\mathbf{T}_L^R \in \mathbf{SE}(3)$ of the stereo camera respectively. For the plane hypotheses, we use all possible permutations drawn from 64 depth values over the range $[0.5,\ 30]\mathrm{m}$ with a constant disparity step size, and fronto-parallel and ground plane orientations. In total, 128 plane hypotheses are sampled. For a particular point $\mathbf{x}_i \in \mathbb{R}^2$ in the left reference image, we define a local patch centered at $\mathbf{x}_i$. All the pixels within the local patch can then be warped to the right image by $\mathbf{H}_L^R$ to form a new warped patch. For each local patch in the left reference image, we compute its corresponding warped patches by all the homography matrices induced by all the plane hypotheses. For each pair of the left reference patch and its right warped patch induced by plane hypothesis $\{\mathbf{n}_m, d_m\}$, we can compute their similarity score by the

Zero mean Normalized Cross Correlation (ZNCC) cost [HHL+15]. We can therefore have 128 possible similarity scores for each sampled pixel in the left reference image. By using winner-takes-all approach, we can obtain the plane hypothesis on which the corresponding 3D point lies by selecting the plane hypothesis $\{\mathbf{n}_m, d_m\}$ which gives the largest similarity score (i.e. the ZNCC score). Its corresponding depth $d_{\mathbf{x}}$ can then be computed by

$$d_{\mathbf{x}} = -\frac{d_m}{\mathbf{n}_f^T \cdot \mathbf{n}_m}, \tag{3.14}$$

where $\mathbf{n}_f^T = [0, 0, 1]$ is a transformed unitary 3D vector.

Stereo matching usually has ambiguities due to repetitive textures, occlusions, and so on. We experimentally find the estimated depth map using a winner-takes-all approach to be rather noisy. To better capture the uncertainty characteristics of the estimated depth, we propose to keep track of multiple depth hypotheses for each pixel. Since the procedure to get these depth hypotheses are the same for all pixels, we illustrate the concept for a particular pixel. For each pixel, we accumulate a 1D (ρ_i, S_i) volume, where ρ_i is the inverse depth (i.e. $\frac{1}{d_x}$ for the i^{th} plane hypothesis by Eq. (3.14)) and S_i is its corresponding ZNCC score. For efficiency, we only select the elements in the volume, such that their ZNCC scores S_i are larger than a pre-defined threshold. We use a threshold value of 0.85 in our experiments (note that a ZNCC score of 1.0 is considered to be perfect). All the selected elements from this volume are then clustered according to ρ_i. In particular, we sort the elements according to ρ_i. We separate two neighboring elements ρ_i and ρ_j into two different clusters if their inverse ray depth difference $|\rho_i - \rho_j|$ is larger than a pre-defined separation distance. We use a cluster separation distance of 0.1. We fit a Gaussian model to each cluster as

$$\mu = \frac{\sum_i S_i \cdot \rho_i}{\sum_i S_i}, \tag{3.15}$$

$$\sigma^2 = \frac{\sum_i S_i \cdot (\rho_i - \mu)^2}{\sum_i S_i}. \tag{3.16}$$

Furthermore, we choose the best score among all S_i belonging to the current cluster as the representative score for the current cluster. We use

the mean value of the cluster with the best score as the inverse depth of the corresponding pixel for direct image alignment. All clusters of each pixel are tracked by temporal motion stereo so that depth ambiguities can be eliminated and the ray depth can be refined iteratively.

3.3.3. Temporal motion stereo

We use temporal motion stereo to refine the estimated depth and to eliminate depth ambiguity in the case that there is more than one inverse depth cluster for each pixel. Due to the epipolar curve issue for fisheye images mentioned earlier, we cannot use the conventional epipolar line disparity search method for stereo matching. Instead, we use a technique to refine the mean depth estimation for each ray segment corresponding to a cluster, which is bounded by a two-sigma distance from the mean inverse depth of that cluster. Since this technique is applied in the same way to each cluster and each pixel, we use one cluster to explain the concept.

Each cluster or each depth hypothesis is parameterized by five parameters: its mean inverse ray depth μ_ρ, the variance of its inverse ray depth σ_ρ^2, its best matching score S_{best}, a good matching count $nInliers$, and a bad matching count $nOutliers$. μ_ρ, σ_ρ^2 and S_{best} are initialized from plane-sweeping stereo. Both $nInliers$ and $nOutliers$ are initialized to be 2 in our experiment to avoid the zero division problem when we compute either the inlier ratio or the outlier ratio. Based on the initial μ_ρ and σ_ρ^2, we assume that the correct inverse ray depth would lie in the interval $[\mu_\rho - 2\sigma_\rho, \mu_\rho + 2\sigma_\rho]$, which has a probability of 95.45% statistically. To get subpixel projection matching accuracy, we subdivide this interval iteratively until the pixel distance between two neighboring projected pixels falls below 1 pixel. For all the sampled inverse ray depths within this interval, we compute their ZNCC matching scores between the reference image and the current image in the same way we compute the scores in plane-sweeping stereo. We use the fronto-parallel and ground plane orientations and current sampled inverse ray depth to compute the homography matrices. Using these homography matrices, we then compute the ZNCC scores between the reference image and its corresponding current image. As in plane-sweeping stereo, each cluster will then accumulate its own

local volume. If this volume is not empty, we fit a new Gaussian model for it and fuse it with the original model by

$$\mu_{refined} = \frac{\frac{\mu_{prev}}{\sigma^2_{prev}} + \frac{\mu_{cur}}{\sigma^2_{cur}}}{\frac{1}{\sigma^2_{prev}} + \frac{1}{\sigma^2_{cur}}}, \tag{3.17}$$

$$\sigma^2_{refined} = \frac{1}{\frac{1}{\sigma^2_{prev}} + \frac{1}{\sigma^2_{cur}}}. \tag{3.18}$$

If no good matching (i.e., $S_{best} < 0.85$) is found, then we increment $nOutliers$ by 1. Otherwise, we increment $nInliers$ by 1. If the outlier ratio

$$\gamma = \frac{nOutliers}{nInliers + nOutliers} \tag{3.19}$$

is larger than a pre-defined threshold, we label the current cluster or depth hypothesis as an outlier and stop tracking it. If only one cluster remains after several motion stereo refinement iterations, and its corresponding variance falls below a certain threshold, we mark the current hypothesis as the true depth estimate and further mark this pixel's depth as converged. If all hypotheses are marked as outliers, we label this pixel as a bad pixel and consider its ray depth estimation to have diverged. Only pixels not labeled as either converged or diverged are refined in the next iteration by new captured images.

3.4. Experimental evaluation

We evaluate our direct fisheye-stereo visual odometry pipeline using real world datasets. In addition, we compare our direct fisheye-stereo visual odometry pipeline against a state-of-the-art semi-direct fisheye-stereo visual odometry algorithm by Heng et al. [HC16]. The evaluation data was collected on a ground vehicle as shown in Fig. 3.2. Fig. 3.3 presents two sample images captured on our ground vehicle. For ground truth data, we use the post-processed pose estimations from a GPS/INS system, which has a position error around 2 cm according to manufacturer specifications.

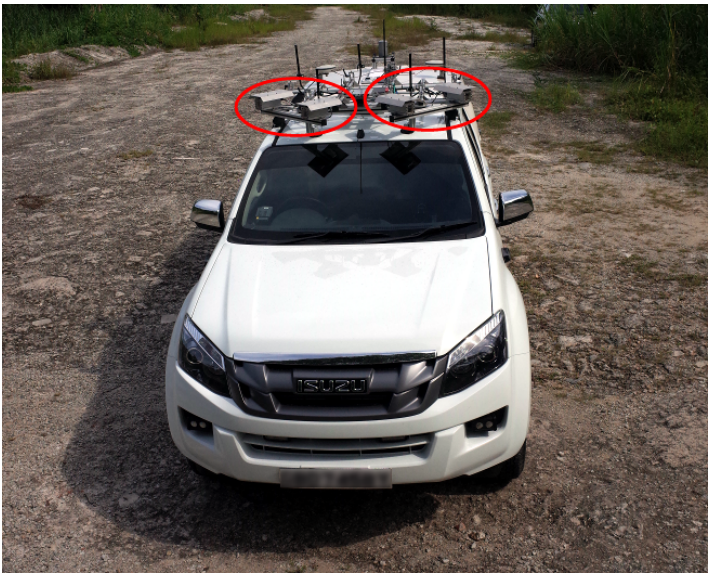


Figure 3.2.: Isuzu D-Max platform equipped with two fisheye-stereo cameras which are shown enclosed in red ellipses.

Figure 3.3.: Sample images of the used dataset for our algorithm evaluations.

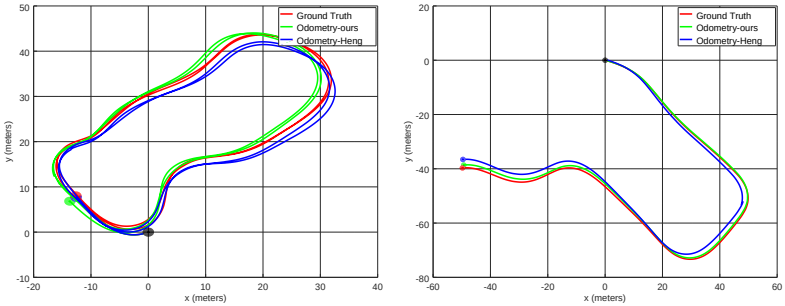


Figure 3.4.: Red, green, and blue lines represent the trajectories estimated by the GPS/INS system, the proposed method, and the method from Heng et al. [HC16] respectively.

We have two fisheye-stereo cameras with 50 cm baseline on our vehicle platform. Fig. 3.2 demonstrates the locations of the fisheye-stereo cameras, which are marked by red ellipses. The left and right fisheye-stereo cameras look $45°$ to the left and right respectively. All cameras output 1280×960 color images at 30 frames per second, and are hardware-time-synchronized with the GPS/INS system. We calibrate the multi-camera system using a grid of AprilTag markers, and use hand-eye calibration to compute the transformation between the reference frames of the multi-camera system and the GPS/INS system. This transformation allows the direct comparison of visual odometry pose estimations with the post-processed GPS/INS pose estimations, which are used as ground truth for evaluation.

Our pipeline is implemented and evaluated on a PC with an Intel i7 CPU @ 2.9 GHz and a low-end Nvidia GTX 480 GPU. We run our tracking thread at 20 Hz on one CPU core. Both plane sweeping stereo and motion stereo run on the GPU at around 5 Hz.

Experiments: To achieve real-time performance, we use the downsampled images (with a resolution of 640×480 pixels) from the left fisheye-stereo camera as input to our direct visual odometry pipeline. The vehicle was driven with a speed range of 10-15 km/h. The trajectory lengths of the used

	xy-drift	yaw-drift
VO-KentRidge-ours	**0.86**%	**0.013** deg/m
VO-KentRidge-Heng	0.96%	0.0138 deg/m
VO-NUS-ours	**0.6**%	**0.0063** deg/m
VO-NUS-Heng	1.6%	0.011 deg/m

Table 3.1.: Pose accuracy comparison between our direct visual odometry algorithm and that from Heng et al. [HC16]

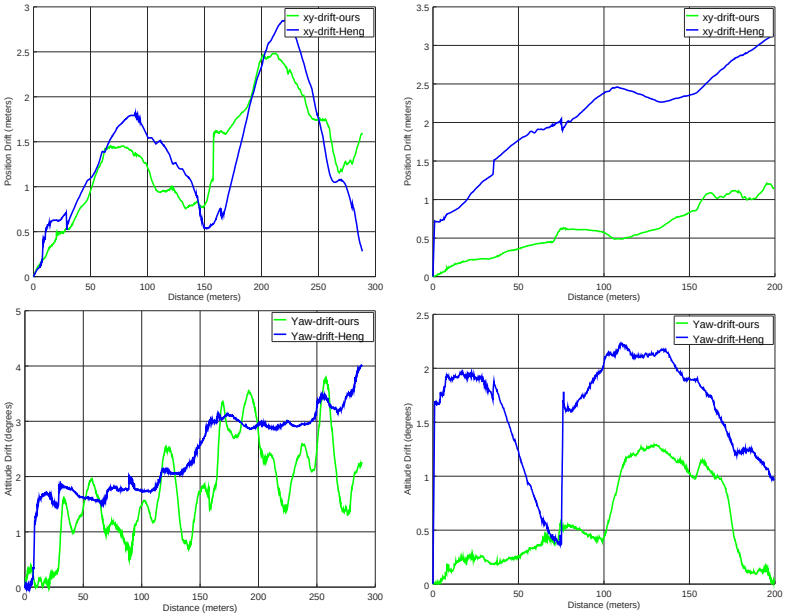


Figure 3.5.: Position errors and orientation errors estimated by our method and the method from Heng et al. [HC16], on the two data sequences captured by our fisheye-stereo camera.

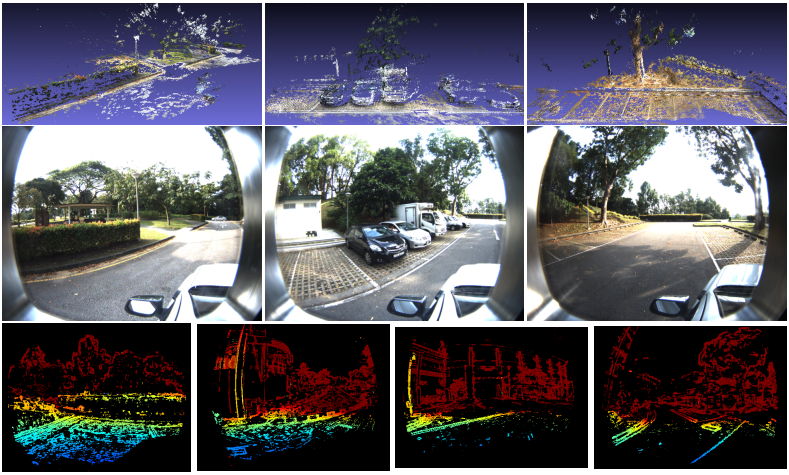


Figure 3.6.: Each row presents the reconstructed point clouds from our VO pipeline, the corresponding images and the estimated semi-dense depth maps.

data sequences are 289 m and 199 m respectively. Fig. 3.4 demonstrates the post-processed pose estimations from the GPS/INS system, our visual odometry algorithm, and the method from Heng et al. [HC16] in red, green and blue respectively. A black circle marks the starting point, while red, green and blue circles mark the end of the trajectories respectively. Fig. 3.5 plots the absolute $x - y$ position errors and the absolute yaw errors against traveled distance respectively. We compute the position error at a given time by computing the norm of the difference between the $x - y$ components of the pose estimates from visual odometry implementations and the GPS/INS system. We compute the yaw error at a given time by computing the absolute difference between the estimated yaw from visual odometry and the GPS/INS system at that time.

We compare the accuracy of our direct visual odometry algorithm against that of Heng et al. [HC16] using position and orientation drift metrics. We

compute the $x-y$ and yaw errors averaged over all possible subsequences of length $\{100, 200\}$ meters. Table 3.1 shows that our direct visual odometry implementation outperforms that of Heng et al. [HC16] for both sequences. In addition, we are able to generate a semi-dense point cloud that can be used for dense mapping. Fig. 3.6 presents the reconstructed and colored point cloud generated by our visual odometry pipeline. The point cloud only includes points which are labeled as converged by our motion stereo. To better show the reconstruction quality of our pipeline, we choose different viewpoints for purposes of comparison. A supplementary video showing the experimental results can also be found at https://youtu.be/NOiIVO0jzuc.

One possible reason that our algorithm performs better than that of Heng et al. [HC16] could be the use of more information from observed images. Our algorithm uses most of the high gradient pixels (>20k) while the method from Heng et al. [HC16] only samples several hundred sparse feature points. The use of more information increases the robustness of the whole algorithm, especially in poorly textured environments where the work of Heng et al. [HC16] does not perform well as shown in Fig. 3.4.

3.5. Conclusion

In this section, we present a direct visual odometry algorithm for a fisheye-stereo camera. The algorithm estimates the camera poses and scene structures directly from raw pixel intensity values, so that the feature detection and matching step can be avoided. We evaluate our method with real world datsets. The experimental results demonstrate that our algorithm is able to estimate the camera motion accuratly. Different from sparse feature based approaches, our algorithm can also provide a semi-dense reconstruction at the same time of pose estimations, which is useful for real-world robotic navigation tasks.

4. Robust VO with a Multi-Camera System

4.1. Introduction

Several approaches have been proposed to improve the robustness of VO for specific environments. For example, Alismail et al. [ABL16b] propose a dense binary descriptor that can be integrated within a multi-channel Lucas Kanade framework to improve illumination change robustness. Park et al. [PSP17] perform a systematic evaluation of real-time capable methods for illumination change robustness in direct visual SLAM. Zhang et al. [ZCS17] propose an active exposure control method for robust visual odometry in high dynamic range (HDR) environments. For each frame, they choose an exposure time that maximizes an image quality metric. In the work from Pascoe et al. [PMT+17], a direct monocular SLAM algorithm based on the Normalized Information Distance (NID) metric is proposed. They show that the information-theoretic NID metric provides robustness to appearance variations due to lighting, weather, and structural changes in the scene. Zhang et al. [GOZGJS18] also propose to take advantage of learning based approach to enhance the image quality for visual odometry in challenging HDR environments.

In this chapter, we propose to improve the robustness of VO by using a multi-camera system, which is mainly based on our publication [LGH+18]. A multi-camera system can cover a wide field-of-view and thus provide redundancy for poorly textured environments. Our algorithm consists of a pose tracker and a local mapper. The tracker estimates the current pose by minimizing photometric errors between the most recent keyframe and the current frame across multiple cameras. The local mapper consists of

a sliding window optimizer and a two-view stereo matcher. Both vehicle poses and structure are jointly optimized by the sliding window optimizer. This optimization minimizes long-term pose drift. The depths of newly sampled feature points are initialized by the two-view stereo matcher. Furthermore, our formulation is designed to be flexible enough to support an arbitrary number of stereo cameras. We thoroughly evaluate our algorithm under multiple challenging conditions. In particular, we select five different datasets with different characteristics for our evaluations. The datasets are captured in varying lighting conditions (e.g. daytime, night-time with near-infrared (NIR) illumination and night- time without NIR illumination), at different vehicle speeds, and over different trajectory lengths. Experimental results demonstrate that a multi-camera setup makes the VO more robust to challenging environments, especially night-time conditions, in which a single stereo configuration fails easily due to the lack of features.

4.2. Notations

We use lower case letters (e.g. λ) for scalar variables, bold lower case letters (e.g. $\mathbf{v}$) for vectors, and bold capital letters (e.g. $\mathbf{T}$) for matrices. A coordinate frame x is denoted as $\mathcal{F}_x$. We define $\mathbf{T}_{B_k}^{W}$ as the transformation matrix that transforms vectors from frame $\mathcal{F}_B$ to frame $\mathcal{F}_W$ at timestamp k. We use $\mathbf{p}^{W_k}$ to denote the vector variable $\mathbf{p}$ represented in frame $\mathcal{F}_W$ at timestamp k. We use $\mathbf{b}_k$ to denote the vector variable $\mathbf{b}$ at timestamp k. Furthermore, we denote $\|\mathbf{x}\|_{\mathbf{\Sigma}}$ as a weighted L_2 norm, i.e., $\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}$.

Coordinate frames: There are three main coordinate frames used throughout the algorithm derivations. They are a global world coordinate frame $\mathcal{F}_W$, a vehicle-centric body coordinate frame $\mathcal{F}_B$, and a camera coordinate frame $\mathcal{F}_{C_i}$ for each camera C_i. Frame $\mathcal{F}_W$ is defined to be fixed relative to the global earth coordinate frame in which the cameras navigate. Furthermore, we define the initial vehicle-centric body coordinate frame $\mathcal{F}_B$ to coincide with $\mathcal{F}_W$, i.e., $\mathcal{F}_B$ at timestamp 0.

Vehicle state: We describe the motion state $\mathcal{S}_k$ at timestamp k by its translation and rotation from frame $\mathcal{F}_B$ to frame $\mathcal{F}_W$. The position is

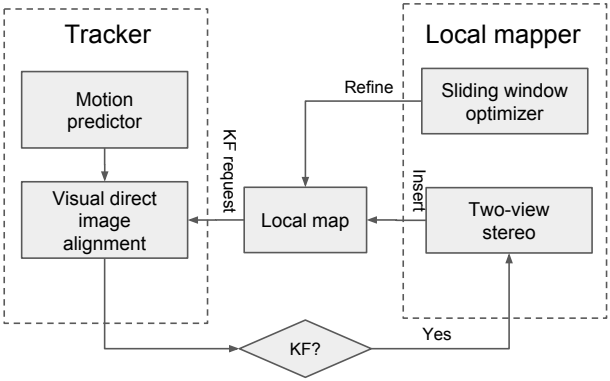


Figure 4.1.: Schematic representation of our VO pipeline.

denoted by $\mathbf{p}^{W_k}$. The orientation is represented by a rotation matrix $\mathbf{R}_{B_k}^{W} \in \mathbf{SO}(3)$. We obtain the homogeneous transformation matrix $\mathbf{T}_{B_k}^{W} \in \mathbf{SE}(3)$ from the vehicle body frame $\mathcal{F}_B$ to the world frame $\mathcal{F}_W$ at timestamp k:

$$\mathbf{T}_{B_k}^{W} = \begin{bmatrix} \mathbf{R}_{B_k}^{W} & \mathbf{p}^{W_k} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} . \tag{4.1}$$

We denote the image captured at timestamp k by camera C_j as $\mathbf{I}_k^{C_j}$. We further denote the measured pixel intensity at pixel coordinates $\mathbf{u}$ in $\mathbf{I}_k^{C_j}$ as $\mathbf{I}_k^{C_j}(\mathbf{u})$; the pixel intensity is a scalar value for grayscale images. For simplicity, we denote the image captured at the latest keyframe as $\mathbf{I}_{KF}^{C_j}$.

4.3. Method

In this section, we describe our VO algorithm for a multi-camera system. Our algorithm is based on the work from Engel et al. [EKC17], which describes a VO framework for a monocular camera. The camera motion is

recovered by direct image alignment algorithm which minimizes a photometric cost function for a sparse set of pixels. We extend the formulation to a multi-camera system which we model as a generalized camera consisting of cameras with multiple centers of projection. As shown in Fig. 4.1, our algorithm consists of two threads: a tracker and a local mapper. The tracker estimates the vehicle pose in real-time using direct image alignment with respect to the latest keyframe. The local mapper is mainly used to minimize long-term pose drift by refining both the vehicle pose and the estimated 3D point cloud. The local mapper uses a computationally intensive batch optimization method for both pose and structure refinement. Thus, it runs at a much lower frame rate compared to the tracker and only keyframes are processed. For a new keyframe, its newly sampled 3D points are initialized by a two-view stereo algorithm. We will describe the algorithm in detail in the following.

Hardware Configurations: To make our algorithm work for different hardware configurations, we assume we have N cameras in total. Each camera can be configured either as a reference camera for motion tracking or as an auxiliary camera for static stereo matching. For ease of reference, we denote the camera C_i as a reference camera as ${}^{r}C_i$. Without loss of generality, we assume that we have N_r reference cameras and N_a auxiliary cameras where $N_r + N_a = N$.

4.3.1. Tracker

As shown in Fig. 4.1, the tracker consists of two parts: a motion predictor and direct image alignment. Direct image alignment is used to estimate the current vehicle pose with respect to the latest keyframe. We use the vehicle pose provided by the motion predictor to initialize the alignment in order to avoid local minima.

Constant velocity motion prediction model: We use a constant motion model to predict the current vehicle pose. Let us define the current vehicle pose as $\mathbf{T}_{B_k}^{W}$ which is the transformation from the vehicle frame $\mathcal{F}_B$ at timestamp k to the world frame $\mathcal{F}_W$. Similarly, we define the vehicle poses corresponding to the two latest frames as $\mathbf{T}_{B_{k-1}}^{W}$ and $\mathbf{T}_{B_{k-2}}^{W}$ respectively.

We assume that the motion velocity from timestamp $k-2$ to k is constant, we thus can predict the current vehicle pose as

$$\mathbf{T}_{B_k}^{W} = \mathbf{T}_{B_{k-1}}^{W}\mathbf{T}_{B_k}^{B_{k-1}} = \mathbf{T}_{B_{k-1}}^{W}\mathbf{T}_{B_{k-1}}^{B_{k-2}} \ , \tag{4.2}$$

where

$$\mathbf{T}_{B_{k-1}}^{B_{k-2}} = (\mathbf{T}_{B_{k-2}}^{W})^{-1}\mathbf{T}_{B_{k-1}}^{W}. \tag{4.3}$$

Direct sparse tracker: Given the initial pose estimate provided by the motion predictor, we use a direct sparse tracker for refinement. In contrast to the local mapper which will be explained in a later section, only images from reference cameras are used for motion tracking. In particular, we estimate the relative vehicle pose $\hat{\mathbf{T}}_{B_{KF}}^{B_k}$ between the latest keyframe and the current frame by minimizing the following energy function:

$$\hat{\mathbf{T}}_{B_{KF}}^{B_k} = \underset{\mathbf{T}_{B_{KF}}^{B_k}}{\operatorname{argmin}} \sum_{i=1}^{N_r} \sum_{\mathbf{u}\in\mathbf{\Omega}(\mathbf{I}_{KF}^{{}^rC_i})} (\mathbf{I}_{KF}^{{}^rC_i}(\mathbf{u}) - \mathbf{I}_{k}^{{}^rC_i}(\hat{\mathbf{u}}))^2 \ . \tag{4.4}$$

Here, N_r is the number of reference cameras and $\mathbf{\Omega}(\mathbf{I}_{KF}^{{}^rC_i})$ is the set of feature pixel points sampled from the keyframe image of reference camera ${}^rC_i$. For each feature point $\mathbf{u}$ in the keyframe image, $\hat{\mathbf{u}}$ is the corresponding pixel position in the current frame, obtained as

$$\hat{\mathbf{u}} = \pi(\mathbf{T}_{B}^{C_i} \cdot \mathbf{T}_{B_{KF}}^{B_k} \cdot (\mathbf{T}_{B}^{C_i})^{-1} \cdot \pi^{-1}(\mathbf{u}, d)) \ . \tag{4.5}$$

$\mathbf{T}_{B}^{C_i}$ is the relative transformation from vehicle body frame to camera frame. π and π^{-1} are the camera projection function and back projection function, respectively. d is the depth of the feature point $\mathbf{u}$ in the keyframe. The only unknown variable is $\mathbf{T}_{B_{KF}}^{B_k}$ since $\mathbf{T}_{B}^{C_i}$ can be obtained from offline sensor calibration and d can be initialized by stereo matching and refined by the joint optimization performed by the local mapper.

The above formulation follows the standard forward compositional method [BM04] which requires the Hessians and Jacobians of the residuals to be computed in every iteration of the optimization. For improved efficiency, we adopt the inverse compositional method [BM04] to minimize

the energy function, as illustrated in the previous chapter. To robustify the least squares estimator which is sensitive to outliers, we further apply a robust loss function (i.e. Huber loss) to all residuals.

Outlier removal: Besides the use of a robust loss function, a specific outlier removal mechanism is also used to robustify both the tracker and the local mapper. We detect outliers based on the template matching scores between the reference frame and the current frame. In our implementation, we use the Zero Mean Normalized Cross-Correlation (ZNCC) score [HHL+15]. If the score is smaller than a predefined threshold, we classify a feature match as an oulier and remove it from the optimization. The tracker already uses image patches around the feature points and the warped image patches are already computed during direct image alignment. Thus, the computational overhead of the outlier removal step is marginal.

Relationship with the tracker from [EKC17]: In contrast to the tracker in the Direct Sparse Odometry (DSO) algorithm [EKC17], we do direct sparse pose tracking. To do pose tracking, DSO projects sampled feature points from all recent keyframes to the current keyframe and then do semi-dense pose tracking. Compared to semi-dense pose tracking, the computational complexity of direct tracking with sparse feature points is far smaller than that of semi-dense tracking. Furthermore, an outlier removal step as described in the previous section is included to make the tracking and mapping more stable.

4.3.2. Keyframe and feature selections

Keyframe selection: One of the implicit assumptions behind motion tracking is that the scene difference between the reference keyframe and the current frame is sufficiently small. If the difference between the current frame and the reference keyframe becomes too large, we instantiate a new keyframe. Intuitively, the difference between frames should be measured based on changes in image content rather than based on absolute pose differences (since the former strongly depends on the depth of the scene while the latter is oblivious to it). Therefore, we use the mean square optical flow as one of the criterions for detecting scene changes. More precisely,

we compute

$$f = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|_2 \quad , \tag{4.6}$$

where $\|\cdot\|_2$ is the Euclidean norm, $\mathbf{u}_i$ is a feature point in the reference keyframe, and $\hat{\mathbf{u}}_i$ is its corresponding feature point in the current frame. If f is above a threshold, we create a new keyframe.

Feature selection: Once a new keyframe is created, sparse feature points are sampled from all reference cameras. We sample N sparse features uniformly from each image based on their gradient magnitudes. In particular, we divide the image into a grid and select the point with the highest gradient magnitude per grid cell. However, we will not sample a point from this grid cell if the highest gradient magnitude of a grid is smaller than a pre-defined threshold.

Feature representation and depth initialization: As per the implementation in [EKC17], we sample all pixels by following a predefined patch pattern (e.g., 5×5 pattern) for each sparse feature. All these pixels are used for motion tracking and local mapping. In particular, the created visual residuals from the tracker and local mapper are per pixel instead of per patch. Furthermore, pixels from the same patch are assumed to lie on the same 3D plane, which can be parameterized by its inverse plane depth and plane normal. The inverse plane depths and plane normals are initialized by a stereo matching algorithm [HHL+15]. The plane depths of the sampled patch instead of ray depths of each pixel are refined during the joint optimization carried out by the local mapper. Each pixel from the same patch has different ray depths, but has the same plane depth. It thus reduces the number of variables to optimize and improves the computational efficiency of the optimizer.

4.3.3. Local mapper

Two-view stereo matching: Stereo matching is used to initialize the depth of each sampled feature from the new keyframe. Rather than performing disparity search along epipolar lines, we use plane-sweeping stereo [HHL+15]

to compute the depths. This allows us to directly operate on the fisheye images, and thus, avoid a loss of field-of-view from having to undistort and rectify the images.

Based on the implementation from [HHL+15], we use the GPU to accelerate plane-sweeping stereo. We sweep planes in two directions: fronto-parallel to the viewing direction of the keyframe and parallel to the ground plane. For each direction, we generate 64 plane hypotheses with a constant disparity step size between them. The planes cover the range $[0.5,\ 30]\ \mathrm{m}$ in front of the camera. This results in a total number of 128 plane hypotheses that are evaluated. We compute the ZNCC score between the 7×7 image patches from $\mathbf{I}^{C_i}$ and warped image patches from $\mathbf{I}^{C_j}$. The plane hypothesis with the largest template matching score is selected for each feature point.

Sliding window optimizer: To minimize drift, the local mapper jointly optimizes all vehicle states and the 3D scene geometry. As the vehicle moves, the number of states increases over time. To bound running time, we use state marginalization to remove old states. We only keep a fixed number of previous states, resulting in a constant computational complexity as proposed by Dong-Si et al. [DSM11]. As demonstrated in the work from Sibley et al. [SMS10], a marginalized state will result in all remaining states connected to it to be connected with each other after marginalization. The resulting Hessian matrix would thus not be sparse anymore. This in turn increases the run-time cost of computing the Schur complement during structure optimization. For efficiency, we follow [LLB+15, EKC17] and only fill terms of the Hessian that do not involve geometry terms. For further efficiency, optimization is only performed on selected keyframes.

In particular, we define a sliding window containing k vehicle states. All states outside this sliding window are treated as old states and are removed through partial marginalization as what has been done in the work of Engel et al. [EKC17]. We use $\mathcal{S}_*$ to represent the set of all vehicle states within the sliding window and they are represented in vector form. The local mapper estimates the vehicle states $\mathcal{S}_*$ inside the sliding window, and the set $\mathbf{D}_*$ of inverse plane depths of all the sampled features within the sliding

window via

$$\hat{\mathcal{S}}_*, \hat{\mathbf{D}}_* = \underset{\mathcal{S}_*, \mathbf{D}_*}{\operatorname{argmin}} \; \mathrm{E}_0(\mathcal{S}_*) + \mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*). \tag{4.7}$$

Here, $\mathrm{E}_0(\mathcal{S}_*)$ is either a prior energy term obtained from partial marginalization or an initial prior and $\mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*)$ is the visual energy term between all keyframes within the sliding window. Optimization is carried out using the Gauss-Newton algorithm. In the following, we describe the individual terms in more detail.

There are two types of **prior terms**. One is from the initial prior. In order to fix the unobservable degrees of freedom of the VO problem, we need to fix the initial state such that all subsequent states are estimated relative to it. Thus, they are usually formulated in the following form:

$$\mathrm{E}_0(\mathcal{S}_0) = \frac{1}{2} \left\| \hat{\mathcal{S}}_0 - \mathcal{S}_0 \right\|_{\mathbf{\Sigma}_0} \quad , \tag{4.8}$$

where $\hat{\mathcal{S}}_0$ is the value of the initial state, $\mathcal{S}_0$ is the inital state variable to estimate and $\mathbf{\Sigma}_0$ is the covariance matrix of the inital state. To fix $\mathcal{S}_0$ as equal to $\hat{\mathcal{S}}_0$, $\mathbf{\Sigma}_0$ is usually selected to have infinitesimal diagonal terms. This prior term only appears in the energy function when the initial state is within the sliding window. Once it leaves the window, it will be marginalized out in the same way as all other energy terms [SMS10, DSM11].

The other prior term is a direct result of partial marginalization. Partial marginalization introduces priors to all remaining states which are connected to the marginalized states [SMS10, DSM11]. The information from eliminated states after their removal is stored in prior Hessian and Jacobian matrices such that we can optimally remove them. Thus, we can have the following prior energy term for the remaining states

$$\mathrm{E}_0(\mathcal{S}_*) = \mathbf{J}_m^T(\hat{\mathcal{S}}_{*0} - \mathcal{S}_*) + \frac{1}{2} \left\| \hat{\mathcal{S}}_{*0} - \mathcal{S}_* \right\|_{\mathbf{H}_m{}^{-1}} \quad , \tag{4.9}$$

where $\hat{\mathcal{S}}_{*0}$ is a set of prior vehicle states of $\mathcal{S}_*$ estimated from previous iterations, $\mathbf{J}_m$ and $\mathbf{H}_m$ are the Jacobian and Hessian matrices respectively, and accumulated from state marginalization.

The **visual energy term** contains the sum of all photometric error residuals and is modeled as

$$\mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*) = \sum_{m=1}^{k} \sum_{i=1}^{N_r} \sum_{\mathbf{u} \in \mathbf{\Omega}(\mathbf{I}_m^{{}^rC_i})} \sum_{n \in \mathbf{\Theta}(\mathbf{u})} \sum_{j \in \mathbf{\Phi}(\mathcal{S}_m, \mathbf{u})} \left\| r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho) \right\|_{\Sigma} , \tag{4.10}$$

Here, k again is the number of keyframes and N_r is the number of reference cameras. ${}^rC_i$ and C_j refer to the i^{th} and j^{th} camera respectively. $\mathbf{\Omega}(\mathbf{I}_m^{{}^rC_i})$ is the set of feature points sampled from the image of the i^{th} reference camera in the m^{th} keyframe. $\mathbf{\Theta}(\mathbf{u})$ is the set of indices of the vehicle states that observe the feature point $\mathbf{u}$. $\mathbf{\Phi}(\mathcal{S}_m, \mathbf{u})$ is the set of indices of the cameras in the m^{th} keyframe that observe feature point $\mathbf{u}$. ρ is the inverse plane depth of the feature $\mathbf{u}$. $r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho)$ is a pixel intensity residual that measures intensity differences between two projections of the same 3D scene point (see below for a definition of the residual). Furthermore, we use both inter-camera and intra-camera irradiance residuals in our implementation. Finally, Σ is the scalar variance of the photometric error residual which is obtained from each camera's photometric calibration.

Let $\mathbf{I}_m^{{}^rC_i}$ be the image captured by the i^{th} camera in the m^{th} keyframe. Consider a feature point $\mathbf{u}$ sampled from that image. Using plane sweeping stereo, we know the inverse depth ρ of its corresponding plane as well as the normal $\mathbf{n}$ of that plane. The 3D point $\mathbf{P_u}$ corresponding to $\mathbf{u}$ is thus given by:

$$\mathbf{P_u} = \frac{1}{\rho \cos\theta} \pi_{{}^rC_i}^{-1}(\mathbf{u}) \ , \tag{4.11}$$

where $\pi_{{}^rC_i}^{-1}$ is the inverse projection function of the camera and $\cos\theta = -\mathbf{n}^T \cdot \pi_{{}^rC_i}^{-1}(\mathbf{u})$ is the angle between the viewing ray corresponding to $\mathbf{u}$ and the plane normal. The vehicle poses $\mathbf{T}_{B_m}^W$ and $\mathbf{T}_{B_n}^W$ at timestamps m and n are initially estimated by the tracker while the relative transformations $\mathbf{T}_B^{C_l}$ between the vehicle frame $\mathcal{F}_B$ and the camera frames $\mathcal{F}_{C_l}$ can be estimated offline. We use these transformations to compute the pixel $\hat{\mathbf{u}}$ in the j^{th} camera and in the n^{th} keyframe corresponding to the feature $\mathbf{u}$ by projecting $\mathbf{P_u}$ into the image. Given $\mathbf{u}$ and $\hat{\mathbf{u}}$, we use their intensity

difference to define the residual $r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho)$:

$$r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho) = \mathbf{I}_m^{{}^rC_i}(\mathbf{p}) - \mathbf{I}_n^{C_j}(\hat{\mathbf{p}}) \ . \tag{4.12}$$

In order to minimize the visual energy term from (4.10), we adjust the motion state parameters (namely the pose $\mathbf{T}_{B_m}^W$) as well as the inverse plane depth of each feature. Rather than looking at a single pixel per feature $\mathbf{u}$, we consider a pixel patch per feature. Each pixel in the patch is warped into the other images and contributes a residual according to Eq. 4.12. In this setting, parameterizing the depth of $\mathbf{u}$ based on the plane has the advantage that we only require a single parameter per patch.

4.4. Experimental evaluation

Hardware Setup: We have installed twelve fisheye cameras on our vehicle platform which is shown in Fig. 4.2. In particular, five cameras are installed at the front of the vehicle, two cameras are installed on each of the left and right sides, and three cameras are installed at the back. We select one stereo pair from each side to evaluate our algorithm. In particular, the front stereo pair has a baseline of 0.722m, the back stereo pair has a baseline of 0.755m, the left stereo pair has a baseline of 0.502m, and the right stereo pair has a baseline of 0.497m. All cameras output 1024×544 gray scale images at 25 frames per second, and are hardware-time-synchronized with the GPS/INS system.

We calibrate the multi-camera system using a grid of AprilTag markers. To compute the transformation between the multi-camera system and the GPS/INS system, we run semi-direct VO [HC16] for each stereo pair, obtain an initial estimate using hand-eye calibration, triangulate landmark points using the GPS/INS poses and feature correspondences from VO, and refine the initial estimate together with the landmark points by minimising the sum of squared reprojection errors while keeping the GPS/INS poses fixed. This transformation allows direct comparison of visual odometry pose estimates with the post-processed GPS/INS pose estimates which are used as ground truth.

Figure 4.2.: Two sample images of the vehicle.

Dataset selection: To avoid parameter overfitting, and, at the same time, evaluate our algorithm thoroughly, we select five datasets with different characteristics for evaluation. The first three datasets are collected in a car-park. The first one (Science-park-day) is collected in normal daylight conditions. The second one (Science-park-night-illum) is collected at night with NIR illumination. The third one (Science-park-night) is collected at night without NIR illumination. The other two datasets are collected from a public urban street. One of them (West-coast-day) is collected in day light conditions. Another one (West-coast-night-no-illum) is collected at night without near-infrared illumination. Three sample images are shown in Fig. 4.3. Furthermore, the characteristics of the datasets are summarized in Table 4.1.

	Length (m)	Max. speed (m/s)
Science-park-day	547.448	3.941
Science-park-night-illum	612.078	3.863
Science-park-night	613.096	3.685
West-coast-day	1179.13	10.68
West-coast-night-no-illum	1224.95	10.23

Table 4.1.: Characteristics of datasets used for evaluation.

Evaluation metrics: We follow the metric used by the KITTI benchmark [GLU12] for accuracy evaluation. In particular, we compute the position and orientation drifts over sequences of length 200m, 400m, 600m and 800m for each frame, and average drifts over all sequences. We take into account the runtime of both the tracker and the mapper as a metric for efficiency evaluation. Furthermore, we observe that the ground truth heights for all science park datasets are not reliable even though a highly accurate GPS/IMU system was used. For example, for the Science-park-day dataset, the closed-loop ground truth height drift is more than 2.5 m. Since these errors introduce significant bias into the evaluation, we only consider horizontal translations of all three science park datasets for accuracy evaluations. The remaining datasets are evaluated with 3-axis translation errors.

Parameter settings: For all experiments, we sample a total of 800 5×5 feature patches. All features are uniformly distributed in each camera. For example, if we consider four stereo pairs, then each stereo pair has 200 features. A sliding window with 5 keyframes is used for the local mapper. The optical flow threshold for keyframe selection is 20 pixels. For the Huber parameter, we use 30 for daylight conditions and 10 for night-time conditions.

Baseline algorithm: We choose ORB-SLAM2 [MAT17b] as the baseline algorithm for our comparisons. Since ORB-SLAM2 does not support the fisheye camera models, we undistort the fisheye images to generate pinhole images for all experiments.

Ablation studies: We conduct thorough ablation studies with respect to the hardware configurations. The results can be found in Table 4.2. It shows that a higher number of stereo cameras improves both the accuracy and robustness of our VO algorithm. In particular, both ORB-SLAM2 [MAT17b] and our single stereo configuration fail easily for all the night sequences when using only a single or two stereo pairs. Based on our observations, the main reason of the failure is due to the lack of good features. However, using a multi-camera setup which covers a larger field of view can provide the required redundancy necessary for handling poorly textured night environments.

Figure 4.3.: Three sample images from our datasets, demonstrating the challenges of the datasets which include strong distortion and absence of features at night.

Datasets	Alg*	F	B	L	R	FB	FL	FR	BR	BL	LR	BLR	FLR	FBL	FBR	FBLR
SD	Ours	0.449	1.266	1.185	1.194	0.56	0.282	0.632	0.86	0.64	1.24	0.511	0.635	0.838	0.524	0.352
	ORB*	1.168	1.565	x	4.203	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SN1	Ours	x	3.7	x	x	x	2.05	x	1.01	1.544	2.265	0.625	0.735	0.956	1.193	0.691
	ORB*	3.24	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SN2	Ours	x	x	x	x	x	x	x	x	x	1.01	1.47	0.815	1.544	1.51	1.03
	ORB*	x	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
WCD	Ours	2.33	1.32	2.17	4.35	0.48	1.79	2.02	3.47	0.96	1.53	1.59	1.97	1.47	0.87	0.9
	ORB*	2.34	1.86	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
WCN	Ours	x	x	x	x	2.88	11.6	8.57	5.12	4.99	2.84	3	2.23	2.33	2.05	1.74
	ORB*	x	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.

Table 4.2.: Accuracy evaluations with offline datasets for different hardware configurations (Accuracy (translational drift) is in units of %, the smaller the better; SD: Science-park-day. SN1: Science-park-night-illum. SN2: Science-park-night. WCD: West-coast-day. WCN: West-coast-night-no-illum. Alg*: Algorithms. ORB*: ORB-SLAM2. F: front stereo pair. B: back stereo pair. L: left stereo pair. R: right stereo pair. x: fails to complete whole sequence. N.A.: not available.)

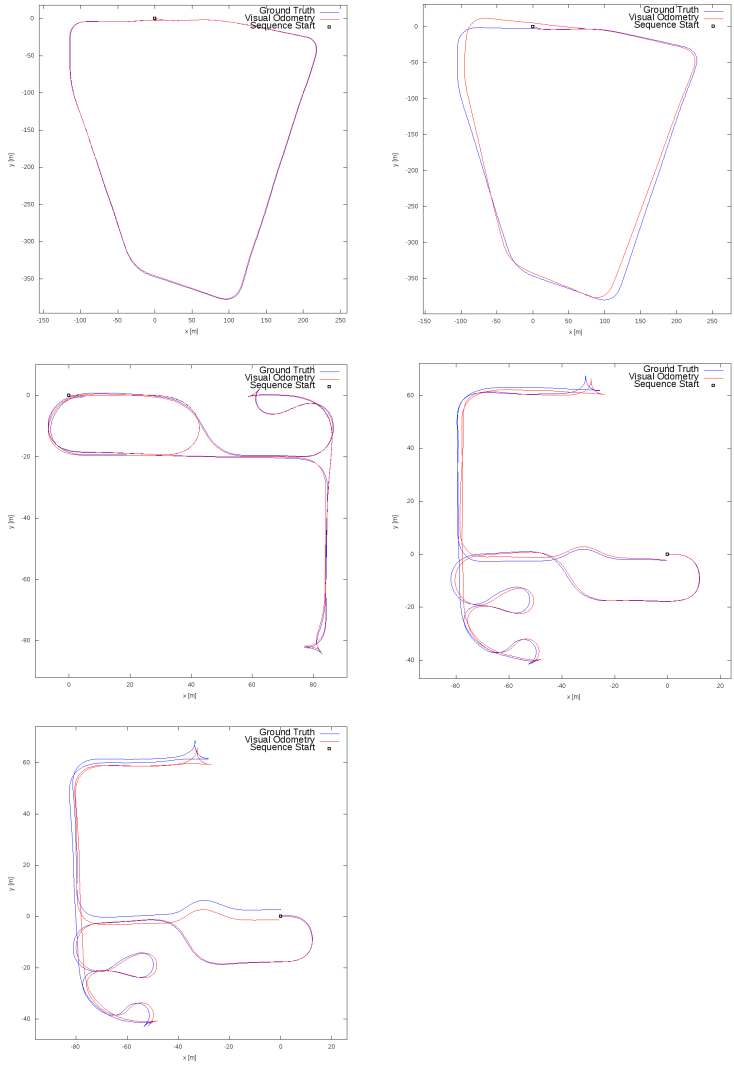


Figure 4.4.: Estimated trajectories of the five selected data sequences.

From Table 4.2, we also observe that our algorithm performs better than ORB-SLAM2 [MAT17b]. We give two possible reasons: (1) the direct method can do refinement with sub-pixel accuracy which improves the accuracy as long as the algorithm is well-implemented, and (2) the image quality degrades after undistortion and stereo rectification, and can affect ORB-SLAM's performance. In contrast, our method is able to directly operate on the raw fisheye input images.

By doing horizontal comparisons (i.e. same dataset) of the same algorithm for different hardware configurations, we can observe that by using more cameras improves the accuracy of our VO algorithm. We think that spatially well distributed features can give better constraints to the optimization problem, which makes our VO algorithm more accurate.

Furthermore, by doing vertical comparisons (i.e. same hardware configuration) of the same algorithm for different light conditions, we can observe that night-time conditions degrade the performance of VO algorithms. Besides the lack of good features, another reason is that the image quality in night-time conditions is worse than that in daylight conditions. When the vehicle moves fast at night, motion blur is inherent in the captured images, especially for the left and right stereo pairs.

Qualitative evaluations: Fig. 4.4 shows qualitative evaluation results of our algorithm for all datasets with four pairs of stereo cameras. We plot the x-y trajectories estimated by our VO algorithm against the ground truth trajectories. A supplementary video can also be found on our project website https://cvg.ethz.ch/research/visual-odometry/.

Runtime efficiency: Our algorithm is evaluated on a laptop with an Intel i7 CPU @ 2.8 GHz. We are able to run our tracking thread at more than 30 Hz and the local mapping thread at around 2 Hz for the current configurations.

4.5. Conclusion

We present a direct sparse visual odometry algorithm for a multi-camera system and robust operation in challenging environments. Our algorithm

includes a direct sparse pose tracker and a local mapper. The tracker tracks the current camera pose in real-time. The local mapper jointly optimizes both poses and structure within a sliding window. Instead of minimizing re-projection errors, both the tracker and mapper directly minimize photometric errors. We evaluate our algorithm extensively with five datasets which have different characteristics. Experimental results show that a multi-camera setup makes the VO more robust to challenging night environments and also improves its accuracy.

Part II.

Deep CNN Enhanced Images

5. Self-supervised Motion Deblurring

5.1. Introduction

Motion blur is one of the most common factors degrading image quality. It often arises when the image content changes quickly (e.g., due to fast camera motion) or when the environment is illuminated poorly, hence necessitating longer exposure times. Combining both situations, e.g., a self-driving car driving at dusk, further aggravates the problem. As many computer vision algorithms such as visual odometry, object detection, or semantic segmentation rely on visual input, blurry images challenge the performance of these algorithms. It is well known that many algorithms (e.g., depth prediction, feature detection, motion estimation, or object recognition) suffer from motion blur [PMB+09, KBM+18, TSN+16, QWMT19]. The motion deblurring problem has thus received considerable attention in the past [SJA08, GJZ+10, NKL17, TGS+18, KBM+18].

Existing techniques to solve this problem can be classified into two categories: the first type of approaches formulate the problem as an optimization problem [KF09, XJ10, FSH+06, SJA08, LWDF09, CL09] where the latent sharp image and/or the blur kernel are optimized using gradient descent. One of the advantages of this kind of methods is that they do not require any ground truth sharp images. However, the resulting solvers usually have a large computational complexity, which limits their applicability in time-constrained settings, such as real-time robotic visual perception. Handcrafted priors on either the image or the blur kernel further limit their performances.

The second type of approaches phrase the task as a learning problem.

Figure 5.1.: **Self-supervised motion deblurring. First:** Sharp ground truth image. **Second:** Blurry input image. **Third:** Deblurring results of our self-supervised method. **Fourth:** Deblurring results of the supervised method from Tao et al. [TGS+18].

Building upon the recent advances of deep convolutional neural networks, state-of-the-art results have been obtained for both single image deblurring [NKL17, TGS+18] and video deblurring [SDW17], outperforming optimization-based techniques in terms of both quality and efficiency. However, learning-based methods typically require full supervision in the form of corresponding pairs of blurred and sharp images. Unfortunately, obtaining such pairs is not always easy due to two main reasons. One is that not every camera has the capability to capture images at enough high frame rate (1000 or more frames per second), such that we can use the recorded frames to synthesize the training data. Another reason is that it would also be difficult to obtain good quality images in real scenarios where the motion blur really occurs (e.g., at night). High frame rate limits the exposure time and would thus make the captured image extremely dark or even invisible.

Inspired by recent progress in self-supervised depth [ZKR+18, GMB17], flow [MHR18, JGR+18] and representation learning [PKD+16, DGE15], we propose a novel approach for self-supervised image deblurring which only relies on real-world blurry image sequences for training in this chapter, which is based on our publication [LJP+20]. Self supervised learning improves the network's generalization performance, by enabling the network to adapt to scenarios where ground truth sharp images are not available. Our network contains a deblurring network and an optical flow

estimation network. However, instead of using ground truth sharp images [NKL17, TGS+18, KBM+18, IMS+17], we pose the task as an inverse rendering problem and take advantage of the physical image formation process for supervision. More specifically, given two consecutive blurry frames of a video sequence, we first predict the corresponding deblurred images using a deep neural network. A second deep neural network takes both deblurred images as input and computes the corresponding optical flow. Using this prediction, and assuming a locally linear blur kernel, our model re-renders the blurred images and compares the results to the original blurry inputs using a photometric loss function. Moreover, we constrain the optical flow network using a photo-consistency loss function. Our entire model can be trained end-to-end from pairs of consecutive blurry images captured with a consumer video camera. At test time, our network takes a single blurred image and deblurs it in real time on a single GTX 1080Ti graphic card using the learned parameters. As illustrated in Fig. 5.1, our approach is competitive with respect to a state-of-the-art supervised method [TGS+18] despite being fully self-supervised.

Our second contribution is a novel synthetic dataset and a real dataset. The synthetic dataset has 3606 blurry-sharp image pairs recorded with a professional high-speed camera mounted on a ground vehicle. The real dataset has 2302 blurry images and is recorded with a normal camera. We use both datasets to evaluate our algorithm against several baselines both quantitatively and qualitatively.

5.2. Method

Fig. 5.2 shows the overall architecture of our model. It comprises four main parts, i.e., the DeblurNet, the FlowNet, the reblur block, and image warping. The DeblurNet is used for single image motion deblurring. It accepts a single blurry image as input and outputs the corresponding sharp image. The two deblurred images are then fed into the FlowNet to estimate a bi-directional dense optical flow field, which will be used to compute spatially varying blur kernels for each blurry image. Given the estimated blur kernels, we reblur the latent sharp image to form a self-consistency

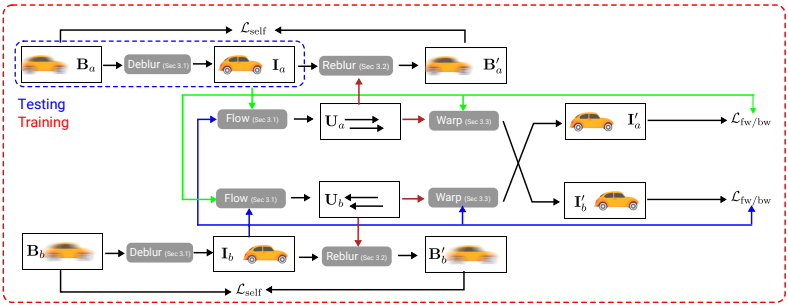


Figure 5.2.: **Architecture of the proposed network.** Given two consecutive blurry images, $\mathbf{B}_a$ and $\mathbf{B}_b$, as input, our network computes the corresponding deblurred images, $\mathbf{I}_a$ and $\mathbf{I}_b$, as well as the bidirectional optical flow, $\mathbf{U}_a$ and $\mathbf{U}_b$. To self-supervise the training of the network, we construct a self-consistency photometric loss $\mathcal{L}_{\text{self}}$ and a forward-backward photometric consistency losses, $\mathcal{L}_{\text{fw/bw}}$.

loss to supervise the training of our network. The FlowNet is trained by maximizing cross-view photometric consistency, which is estimated by image warping. While our approach uses two images for training, the DeblurNet only uses a single input image. After training, our method can thus be used for single image deblurring. The whole network is trained end-to-end without using any ground truth data in the form of sharp images or optical flow. We will now present all components of our model (i.e., the deblurring, optical flow, reblurring and image warping components as well as the loss functions) in detail.

5.2.1. Deblurring and optical flow

For the deblurring and optical flow modules, we take advantage of existing neural network architectures which have performed well in the past for the respective supervised learning tasks [NKL17, KBM+18, TGS+18, SYLK18, IMS+17]. In particular, we adopt the single image deblurring

network from Tao et al. [TGS+18] and the dense optical flow estimation network PWC-Net from Sun et al. [SYLK18]. We make the following modifications for the deblurring network for our particular problem: 1) We replace the deconvolution layer with bilinear upsampling followed by a 3x3 convolution to avoid upsampling artifacts. 2) We add one more Encoder-Decoder block to increase the capacity of the network. 3) We train the network at a single scale without using the LSTM layer to improve both the training and test efficiency. The resulting network is more efficient than the original network while keeping similar deblurring performance. The detail of the network is shown in Fig. 5.3.

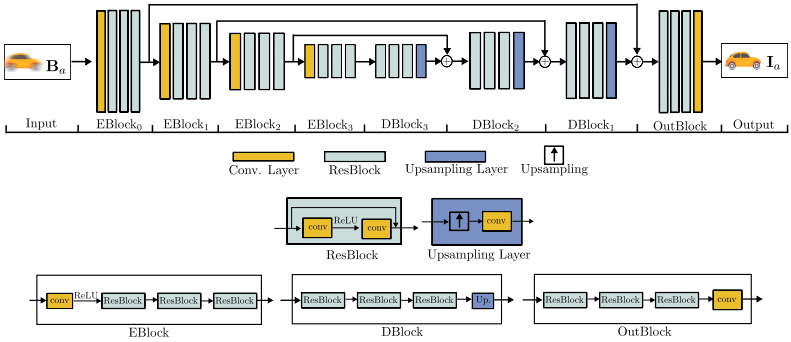


Figure 5.3.: **Architecture of the deblurring network.** Given the input blurry image $\mathbf{B}_a$, the deblurring network outputs the deblurred latent sharp image $\mathbf{I}_a$. Best viewed in enlarged digital version.

5.2.2. Reblurring

The reblurring module encapsulates the physical image formation process, which blurs a sharp image based on the optical flow. Digital cameras operate by collecting photons during the time of exposure and converting those into measurable charge. This process can be formalized by considering the blurred color image $\mathbf{B} \in \mathbb{R}^{W \times H \times 3}$ as the result of integrating virtual sharp images $\mathbf{I}_t \in \mathbb{R}^{W \times H \times 3}$:

$$\mathbf{B}(\mathbf{x}) = \lambda \int_0^\tau \mathbf{I}_t(\mathbf{x})\mathrm{dt} \approx \frac{1}{2N+1} \sum_{i=-N}^{N} \mathbf{I}_i(\mathbf{x}) \tag{5.1}$$

Here, λ is a normalization factor, τ is the exposure time, $\mathbf{x} \in \mathbb{R}^2$ represents the pixel location, $\mathbf{B}(\mathbf{x})$ denotes the motion blurred image at pixel $\mathbf{x}$, and $\mathbf{I}_t(\mathbf{x})$ is the virtual sharp image at pixel $\mathbf{x}$ and time t. The continuous integration can be approximated by using a finite sample size of $2N+1$ virtual sharp frames $\mathbf{I}_i$. We denote the central reference frame, which is the latent sharp image to be estimated, as $\mathbf{I}_0$.

As the exposure time τ is typically small (<200 ms), we may assume that during the time of exposure the image content is primarily affected by image motion and not by other changes like object appearance or illumination. We thus model the virtual sharp frames $\mathbf{I}_i$ as the result of the sharp central reference frame $\mathbf{I}_0$ warped by optical flow $\mathbf{u}_{i\to 0}$:

$$\mathbf{I}_i(\mathbf{x}) = \mathbf{I}_0(\mathbf{x} + \mathbf{u}_{i\to 0}) \tag{5.2}$$

Here, $\mathbf{u}_{i\to 0} \in \mathbb{R}^2$ denotes the optical flow from virtual image $\mathbf{I}_i$ to reference image $\mathbf{I}_0$ at pixel $\mathbf{x}$. Thus, we can reformulate (5.1) as

$$\mathbf{B}(\mathbf{x}) \approx \frac{1}{2N+1} \sum_{i=-N}^{N} \mathbf{I}_0(\mathbf{x} + \mathbf{u}_{i\to 0}) \tag{5.3}$$

and estimate $\mathbf{I}_0$ as well as the optical flow fields $\mathbf{U}_{i\to 0}$ instead of all virtual frames $\mathbf{I}_i$ for solving the deblurring problem. However, the problem is still severely underconstrained as we would need to estimate one optical flow per frame $i \in \{-N, \ldots, N\}$.

We therefore further simplify the model by assuming linear motion during the time of exposure. This is a reasonable assumption in many scenarios where the exposure time is comparably small and rapid motion changes during this time are prevented by the mass and inertia of physical objects (e.g., when the camera is mounted to a vehicle).

Let $\mathbf{u} \in \mathbb{R}^2$ denote the optical flow from frame $\mathbf{I}_0$ to frame $\mathbf{I}_1$ at pixel $\mathbf{x}$. We will demonstrate how to obtain $\mathbf{u}$ in the following section. Assuming

linear motion and equidistant time steps, we obtain the optical flow from frame 0 to frame i as

$$\mathbf{u}_{0\to i} = i \cdot \mathbf{u}. \tag{5.4}$$

Note that in this model the direction of the optical flow is reversed compared to (5.3). We must therefore apply forward warping to obtain the virtual sharp images $\mathbf{I}_i$. This yields

$$\mathbf{B}(\mathbf{x}) \approx \frac{1}{2N+1} \sum_{i=-N}^{N} (\mathcal{W}_{0\to i} \circ \mathbf{I}_0)(\mathbf{x}) \tag{5.5}$$

where the operator $\mathcal{W}_{0\to i}$ warps the reference frame $\mathbf{I}_0$ into the virtual frame $\mathbf{I}_i$ based on the interpolated flow $\mathbf{u}_{0\to i}$. We next describe our implementation of the forward warping operator $\mathcal{W}_{0\to i}$. Note that this operator needs to be differentiable as both the reference frame $\mathbf{I}_0$ and the optical flow $\mathbf{U}$ are outputs of neural networks.

We first construct a regular triangular lattice from the pixel grid by connecting vertices from adjacent pixels as shown in Fig. 5.4 for an example image of size 3×3 pixels. We then warp each vertex of this lattice according to the optical flow $\mathbf{u}_{0\to i}$. The intensities of $\mathbf{I}_i$ (i.e., of the blue pixels) are obtained by linear interpolation [1]. Consider the red pixel $\mathbf{x}$ in Fig. 5.4 as an example. Let further $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$ denote the positions of the vertices belonging to the triangle which covers the red pixel. Then, $\mathbf{I}_i$ is obtained as

$$\mathbf{I}_i(\mathbf{x}) = \omega_0 \mathbf{I}_0(\mathbf{x}_0) + \omega_1 \mathbf{I}_0(\mathbf{x}_1) + \omega_2 \mathbf{I}_0(\mathbf{x}_2) \tag{5.6}$$

where ω_0, ω_1 and ω_2 denote the barycentric coordinates of the point in the triangle. The synthesized motion blurred image can then be computed as the average of all the warped frames as described in (5.1). In the case of occlusions, i.e., when multiple triangles overlap a single pixel, we consider the triangle with the largest motion to be in front. This is a commonly used heuristics [KUH18] which often holds true in practice (in particular when image motion is dominated by camera motion). Note that due to

[1] Note that bilinear interpolation cannot be applied since the warped grid might not be rectangular due to the non-uniform optical flow, as shown in Fig. 5.4.

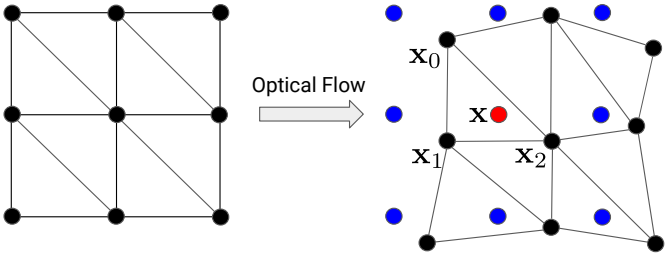


Figure 5.4.: **Differentiable forward warping.** We construct a regular triangular lattice from the pixel grid of the reference image $\mathbf{I}_0$ (left). We then warp each vertex of this lattice according to the optical flow $\mathbf{u}_{0\to i}$. The intensities of $\mathbf{I}_i$ (i.e., of the blue pixels) are obtained by linear interpolation.

the linear interpolation, the warping function $\mathcal{W}_{0\to i}$ is piecewise smooth. As illustrated in Fig. 5.2, our model is symmetric, thus we reblur both the first and the second frame and compare the reblurred result to the original blurry images using a photoconsistency loss. We will use $\mathbf{B}'$ to denote the reblurred image and $\mathbf{B}$ to denote the blurred input image in the following.

5.2.3. Image warping

We found that a photoconsistency loss on the reblurred images alone is insufficient to constrain the optical flow. We thus add an additional self-supervised photometric loss on the optical flow as proposed in prior work [JGR+18, MHR18, WBZL18] and detailed in Section 5.2.5. The input to this loss function is the deblurred image and the deblurred image from the other frame warped based on the estimated optical flow. To warp the images into each other, we exploit backward warping as the optical flow in both directions is known. Let $\mathbf{I}_a \equiv \mathbf{I}_0^a$ and $\mathbf{I}_b \equiv \mathbf{I}_0^b$ denote the first and the second deblurred image, and let $\mathbf{u}_{a\to b}$ and $\mathbf{u}_{b\to a}$ denote the optical flow

between them. The warped deblurred images are obtained as

$$\mathbf{I}'_a(\mathbf{x}) = \mathbf{I}_b(\mathbf{x} + \mathbf{u}_{a\to b}) \tag{5.7}$$
$$\mathbf{I}'_b(\mathbf{x}) = \mathbf{I}_a(\mathbf{x} + \mathbf{u}_{b\to a}) \tag{5.8}$$

using bilinear interpolation [JSZK15]. Note that no triangular mesh needs to be constructed during backward warping.

5.2.4. Relationship between $\mathbf{u}_{a\to b}$/$\mathbf{u}_{b\to a}$ and $\mathbf{u}$

In this section, we present the relationship between $\mathbf{u}_{a\to b}$/$\mathbf{u}_{b\to a}$ and $\mathbf{u}$. $\mathbf{u}_{a\to b}$ and $\mathbf{u}_{b\to a}$ are the bidirectional dense optical flows between the latent sharp images $\mathbf{I}_a$ and $\mathbf{I}_b$, respectively. We assume the motion between $\mathbf{I}_a$ and $\mathbf{I}_b$ to be linear. Without loss of generality, we assume $\mathbf{u}$ is used to synthesize the first blurry image $\mathbf{B}_a$ (i.e. details can be found from Eq. (5.4)). Thus, we obtain $\mathbf{u}$ as the flow from the central virtual frame $\mathbf{I}_0$ to the first virtual image $\mathbf{I}_1$ by linearly scaling $\mathbf{u}_{a\to b}$ according to

$$\mathbf{u} \approx \frac{\tau_a}{2\mathrm{N}\Delta t}\mathbf{u}_{a\to b} \ , \tag{5.9}$$

where τ_a is the exposure time of $\mathbf{B}_a$, $2\mathrm{N}+1$ is the number of sampled virtual sharp frames to synthesize $\mathbf{B}_a$, and Δt is the time interval between $\mathbf{I}_a$ and $\mathbf{I}_b$. Similarly, if $\mathbf{u}$ is used to synthesize the second blurry image $\mathbf{B}_b$, we get

$$\mathbf{u} \approx \frac{\tau_b}{2\mathrm{N}\Delta t}\mathbf{u}_{b\to a} \ , \tag{5.10}$$

where τ_b is the exposure time of $\mathbf{B}_b$.

5.2.5. Loss functions

Our network comprises two types of losses: a self-consistency loss $\mathcal{L}_{\text{self}}$ and a forward-backward consistency loss $\mathcal{L}_{\text{fw/bw}}$. The self-consistency loss

$$\mathcal{L}_{\text{self}} = \|\mathbf{B}'_a - \mathbf{B}_a\|_1 + \|\mathbf{B}'_b - \mathbf{B}_b\|_1 \tag{5.11}$$

penalizes differences between the synthesized motion blurred images $\mathbf{B}'_a$, $\mathbf{B}'_b$ and the original blurred inputs $\mathbf{B}_a$, $\mathbf{B}_b$ using a ℓ_1 loss function. Similarly, the forward-backward consistency loss

$$\mathcal{L}_{\text{fw/bw}} = \|\mathbf{I}'_a - \mathbf{I}_a\|_1 + \|\mathbf{I}'_b - \mathbf{I}_b\|_1 \tag{5.12}$$

penalizes differences between the warped deblurred images $\mathbf{I}'_a$, $\mathbf{I}'_b$ and the estimated deblurred images $\mathbf{I}_a$, $\mathbf{I}_b$. The final loss is a weighted combination

$$\mathcal{L} = \mathcal{L}_{\text{self}} + \lambda \mathcal{L}_{\text{fw/bw}}, \tag{5.13}$$

where λ is a hyper-parameter to balance both losses.

5.2.6. Occlusion handling

As occlusions affect the training of our network, especially at image boundaries, we detect occluded image regions and mask the loss functions (5.11) and (5.12) accordingly. We follow the method used in [WBZL18] to detect occluded image regions. More specifically, we compute the non-occluded regions in $\mathbf{I}_a$ by following the optical flow $\mathbf{U}_{b\to a}$ from $\mathbf{I}_b$ to $\mathbf{I}_a$. We consider all pixels of $\mathbf{I}_a$ which can be reached from $\mathbf{I}_b$ via $\mathbf{U}_{b\to a}$ as non-occluded. Similarly, we can also compute a mask for each virtual frame by following the optical flow from the central image to the virtual frame $\mathbf{u}_{0\to i}$. Since the synthesized blurry image is computed as the average of these virtual frames, we compute the final mask for $\mathcal{L}_{self}$ as the product of all masks for the virtual frames.

5.2.7. Differences with the method proposed by Chen et al.

The overall structure of our method is similar to the work from [CGG+18]. However, we are different in the following two key aspects: **1)** In order to achieve state-of-the-art performance, [CGG+18] uses a supervised loss (section 3.4 from [CGG+18]). [CGG+18] is thus actually a supervised method. **2)** The core component of both methods, i.e., the reblurring module, is different. In fact, blurring a sharp image using convolutions as

done in [CGG+18] is physically incorrect (section 3.3 from [CGG+18]) and only holds for spatially uniform blur. We will make the differences to [CGG+18] more clear.

For simplicity, let us assume we have a one dimensional sharp image $\mathbf{I}$ with N pixels. We further assume the blur kernel corresponds to pixel $\mathbf{I}_i$ as $\{-2, 2\}$ in the form of bidirectional 1D flow. Using the definition from [CGG+18], the convolution based model results in a blurred image of $\mathbf{I}_i$ as $\mathbf{B}_i = \frac{1}{5}\sum_{j=-2}^{2}\mathbf{I}_{i+j}$. A blur kernel $\{-2, 2\}$ of $\mathbf{I}_i$ means that $\mathbf{I}_i$ will contribute to $\mathbf{B}_{i-2}, \mathbf{B}_{i-1}, \mathbf{B}_i, \mathbf{B}_{i+1}, \mathbf{B}_{i+2}$ physically, in contrast to that $\mathbf{I}_{i-2}, \mathbf{I}_{i-1}, \mathbf{I}_i, \mathbf{I}_{i+1}, \mathbf{I}_{i+2}$ will contribute to $\mathbf{B}_i$ as what the convolution based model in [CGG+18] does. Our model eliminates this problem by forward warping the sharp image $\mathbf{I}$ by a fraction of the blur kernels at each sampled timestamp. The blurred image is computed by averaging all these forward warped sharp images to simulate the real motion blurring image formation process. Experimental results shown later demonstrate that the algorithm relying on convolution based model exhibits ringing artifacts on egde boundaries, which degrade the deblurred images.

5.3. Experimental evaluation

Datasets: The dataset from [NKL17] is commonly used to benchmark single image motion deblurring algorithms. It is collected from a hand-held camera. Stronger blur was artificially created by shaking the camera during the recordings. It results in very non-linear camera motions, which violates our motion assumption. Therefore, we collected a new large dataset using a professional Fastec TS5[2] high speed camera mounted on a car. The dataset consists of 196 sequences in total, which are collected at 1200 fps with VGA resolution in diverse environments. The motion blurred images are generated by averaging several consecutive frames (i.e., 1∼50 frames) to simulate the real physical image formation process. To reduce the redundancy per image sequence, we limit the maximum number of blurry-sharp image pairs to 20 per sequence, which results in a total of

[2]https://www.fastecimaging.com/fastec-high-speed-cameras-ts-series/

3606 pairs. We split the dataset into 157 training sequences and 39 test sequences, which results in 2820 image pairs for training and 786 image pairs for evaluation.

We also collect a real motion blurry dataset with 2302 images. The camera is mounted on a tram and captures images at around 50 FPS with a resolution of 752×480 pixels. The dataset is collected at late afternoon and night, when the motion blur would really occur. We split 2062 images for training and 240 images for test.

Implementation details: We implemented our network by using PyTorch [PGC+17]. We empirically set the hyper-parameter λ to be 2.0. To better initialize the network, we pretrain both the DeblurNet and PWC-Net on the blurry images. In particular, we pretrain the DeblurNet for 30 epochs to learn the identity mapping from blurry image to blurry image. We pretrain the PWC-Net for 200 epochs with the blurry sequences in a self-supervised manner. The learning rate used for both networks is 10^{-4}. The whole network is then trained jointly for another 500 epochs, with a learning rate of 10^{-4} for the first 260 epochs and then decayed by half every 40 epochs.

Baselines and experimental settings: We compare the single image deblurring results of our network quantitatively and qualitatively with a state-of-the-art optimization-based method [XZJ13], supervised methods [NKL17,TGS+18,KBM+18] as well as the domain specific self-supervised method from [MKA18]. We train all networks with their recommended hyperparameter settings on our synthetic dataset. For the optimization-based method from [XZJ13], we increase the blur kernel size to 10 pixels to account for the large motions present in our dataset.

Evaluation metrics: We use the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index (SSIM) measures commonly used in the community [SJA08, NKL17, TGS+18] to evaluate the quality of the deblurring results. Larger PSNR/SSIM values indicate better image quality. The efficiency of the methods is evaluated by their total time consumption, but excluding the image loading and saving time.

Ablation studies on the modified DeblurNet architecture: As discussed in Sec.5.2.1, we did several improvements to the original network from [TGS+18]. To evaluate the efficacy of the new network, we train

Network	PSNR↑	SSIM↑	Time ↓
SRN-Deblur [TGS+18]	34.64 dB	0.93	0.13 s
Ours (supervised)	35.04 dB	0.94	0.05 s

Table 5.1.: **Single image deblurring comparison on the synthetic dataset.** We compare our modified network with the original network from Tao et al. [TGS+18] by training both in a supervised way.

both networks in a supervised manner on our synthetic dataset. Table 5.1 presents the comparisons when evaluated under the same settings. It demonstrates our DeblurNet is more efficient than the original network while has slightly better deblurring performance.

Ablation studies on the self-supervision loss for the flow network: To better understand the proposed algorithm, we perform an ablation study on the necessity to train the flow network in a self-supervised manner. We train our proposed network with and without the self-supervision loss for the flow network. The officially provided pretrained model on FlyingChair dataset [ISKB18] is used if the self-supervision loss is disabled. Experimental results demonstrate that the flow network pretrained on the FlyingChair dataset [ISKB18] can generalize to our dataset, but with limited performance. The resulting deblur network gives a PSNR metric as 31.23dB and a SSIM metric as 0.89 on our synthetic dataset, in contract to 32.24dB/0.91 if the network is trained in a fully self-supervised manner. It proves it is beneficial to train the flow network with a self-supervision loss.

Ablation studies on our proposed reblur model: As discussed in Sec.5.2.7, our ablation study supports our claim about the difference between the reblurring modules. For fair comparisons, we trained both the network with convolution based reblur model and the network with our physically correct reblur model under the same settings in an unsupervised fashion. The network with convolution based image formation model yields a PSNR metric of 27.22dB and a SSIM metric of 0.8 on our synthetic

Figure 5.5.: **Qualitative comparisons on synthetic dataset.** **First**: Ground truth sharp image. **Second**: Input blurry image. **Third**: Deblurring results of the supervised method from Tao et al. [TGS+18]; The network is retrained on our dataset. **Fourth**: Deblurring results of the proposed self-supervised learning method.

dataset, while ours yields PSNR and SSIM metrics as 32.24dB and 0.91 respectively.

The necessity to do self-supervised motion deblurring: In real scenarios, motion blur usually occurs in bad illumination conditions. In these cases, it impedes the acquisition with low shutter times to obtain sharp images for supervised learning. One way to address this problem is to train a network with datasets collected under good illumination conditions and transfer the model to scenarios, where motion blur would occur. However, the generalization ability is still questionable due to the large difference between the image textures for both scenarios. We thus evaluate the generalization performance quantitatively and qualitatively with both our synthetic dataset and real dataset respectively. Note that it is not easy to obtain ground

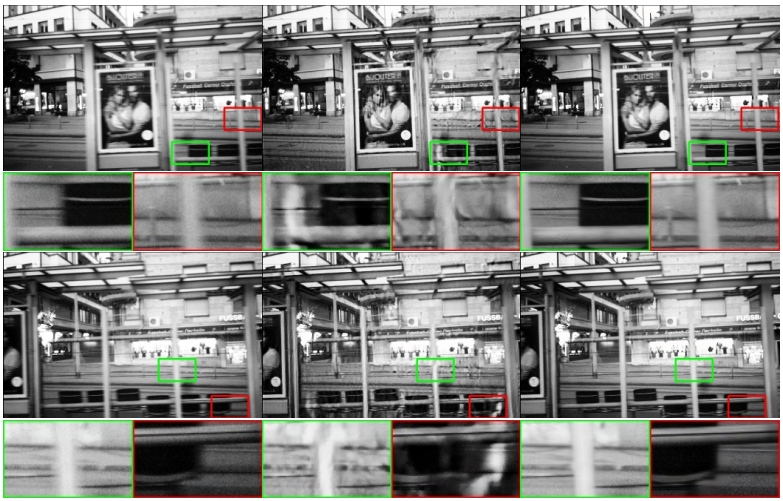


Figure 5.6.: **Qualitative evaluations on real dataset. First** Blurry image. **Second** Deblurred image by the official pretrained network from Tao et al. [TGS+18]. **Third** Deblurred image by our method. The images are post processed for better visualization. Best viewed in digital version.

truth sharp images in real scenarios. We apply the pretrained networks on our test data directly, to evaluate the generalization performance. Table 5.2 and Fig. 5.6 present the experimental results. It demonstrates that all the baseline networks have limited generalization ability and perform worse than our method with a large margin. It proves that self-supervision is beneficial for the network to adapt to scenarios, where the ground truth data is difficult to obtain.

Quantitative and qualitative evaluations on synthetic dataset: Table 5.2 and Fig. 5.5 show quantitative and qualitative comparisons on the synthetic dataset. For the qualitative results, we only compare against the best supervised method [TGS+18].

As can be seen in Table 5.2, our method outperforms both [XZJ13] and [MKA18] significantly in terms of PSNR and SSIM. For optimization-based single image deblurring algorithms (e.g., [XZJ13]), they usually assume the motion blur is caused by either camera rotation or in-plane translation. However, this assumption is violated in our setting for a self-driving scenario. Thus, [XZJ13] leads to poor performance on our dataset. [MKA18] is designed for simple domain-specific blurry images, such as text and facial images. Therefore, it struggles on our dataset that exhibits complex real-world challenges which are harder to learn. In comparison to supervised methods, our method demonstrates competitive results in our quantitative and qualitative evaluation. As expected, there is still a gap between our method and the supervised methods if the ground truth sharp images are available. However, our method outperforms them with a large margin if they are pretrained on other datasets. It demonstrates that self-supervision enables the network to generalize better to real scenarios, where the ground truth data is usually difficult to obtain. It also demonstrates that our method is amongst the fastest methods and can run in real time on a single GTX1080Ti Graphic card.

Qualitative evaluations on real dataset: Since we do not have ground truth sharp images in our real dataset, we cannot refine the baseline networks on it. We thus use the official pretrained networks for the experiments. Fig. 5.6 demonstrates that our method can successfully deblur the blurry images, while the pretrained network from Tao et al. [TGS+18] results in images with artifacts.

Additional qualitative experimental results: In Fig. 5.7 to Fig. 5.13, we present additional qualitative experimental results on single image deblurring on the synthetic dataset. The results demonstrate that our method can generate visually compelling sharp images that are competitive to three state-of-the-art supervised methods [NKL17, KBM+18, TGS+18]. For fair comparisons, we retrain all the networks on our Fastec dataset. It also significantly outperforms the state-of-the-art optimization-based method from Xu et al. [XZJ13] and the self-supervised method from [MKA18].

To further demonstrate the temporal consistency of our method, we also present the experimental results for image sequences from Fig. 5.14

	Method	PSNR↑	SSIM↑	Time ↓
Opt.-based	Xu et al. [XZJ13]	26.04 dB	0.78	377.8 s
Supervised -retrained	DeepDeblur [NKL17]	33.55 dB	0.92	3.45 s
	DeblurGAN [KBM+18]	33.23 dB	0.91	0.06 s
	SRN-Deblur [TGS+18]	34.64 dB	0.93	0.13 s
Supervised -pretrained	DeepDeblur [NKL17]	29.91 dB	0.87	3.45 s
	DeblurGAN [KBM+18]	28.70 dB	0.88	0.06 s
	SRN-Deblur [TGS+18]	30.71 dB	0.88	0.13 s
Self-supervised	Madam et al. [MKA18]	21.69 dB	0.75	0.25 s
	Ours	32.24 dB	0.91	0.05 s

Table 5.2.: **Single image deblurring on synthetic dataset.** Supervised-retrained denotes we retrained the networks with our training data. Supervised-pretrained denotes we use the official pre-trained models to evaluate on our test data directly.

to Fig. 5.19. The experimental results demonstrate that our network can deblur an image sequence temporally consistent, on both the synthetic and real datasets.

5.4. Conclusion

In this chapter, we have presented a self-supervised learning algorithm for image deblurring. Instead of using ground truth sharp images, we leverage the geometric constraints between two consecutive blurry images to supervise training of our network. Both the latent sharp image and motion blur kernel are estimated by a deblur network and an optical flow estimation nework, respectively. Experimental results show that the proposed algorithm outperforms the previously self-supervised method and can produce competitive results compared to supervised methods. It further demonstrates that our method can be trained with real motion blurry data and generalizes well to real unseen data.

Figure 5.7.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

Figure 5.8.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

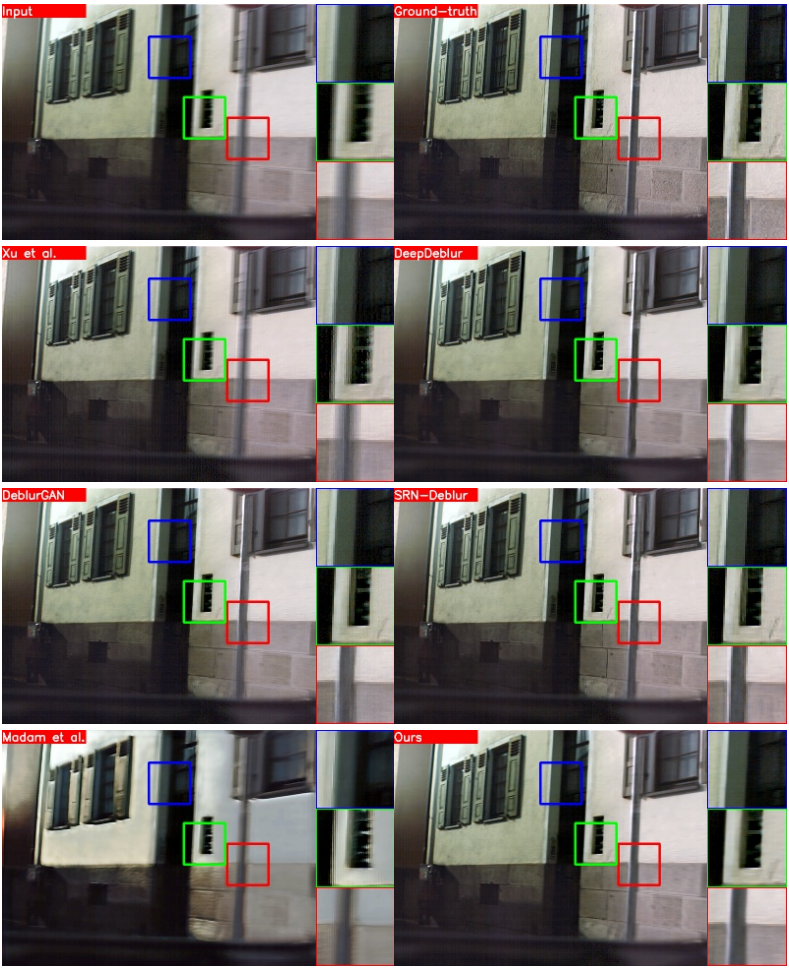


Figure 5.9.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

Figure 5.10.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

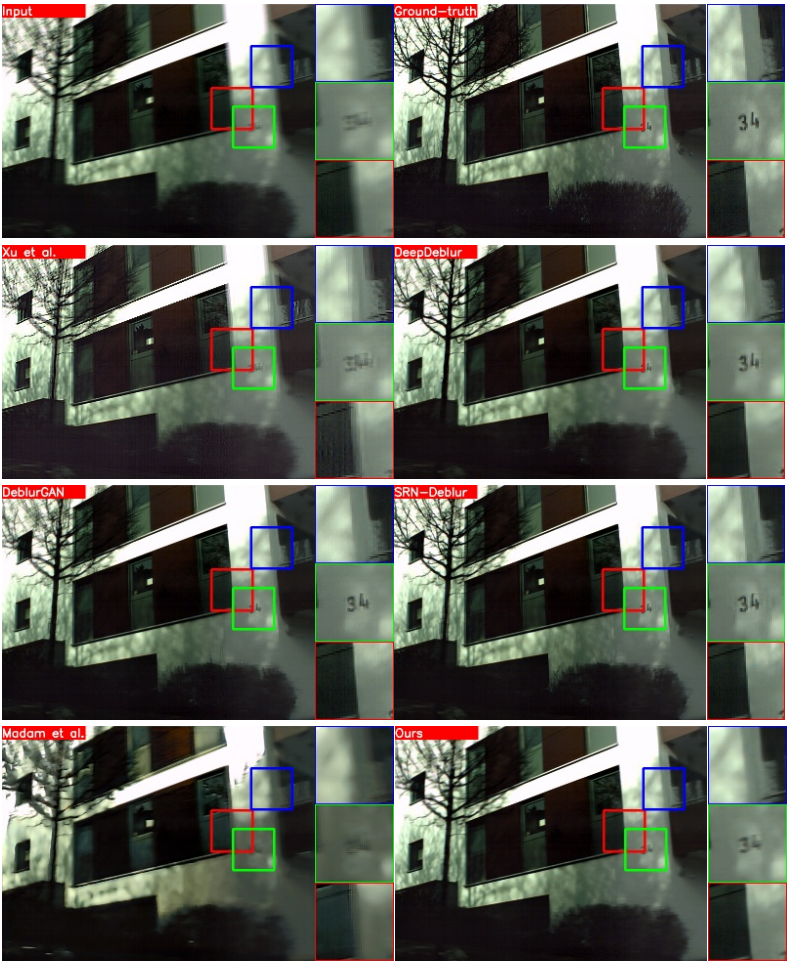


Figure 5.11.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

Figure 5.12.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

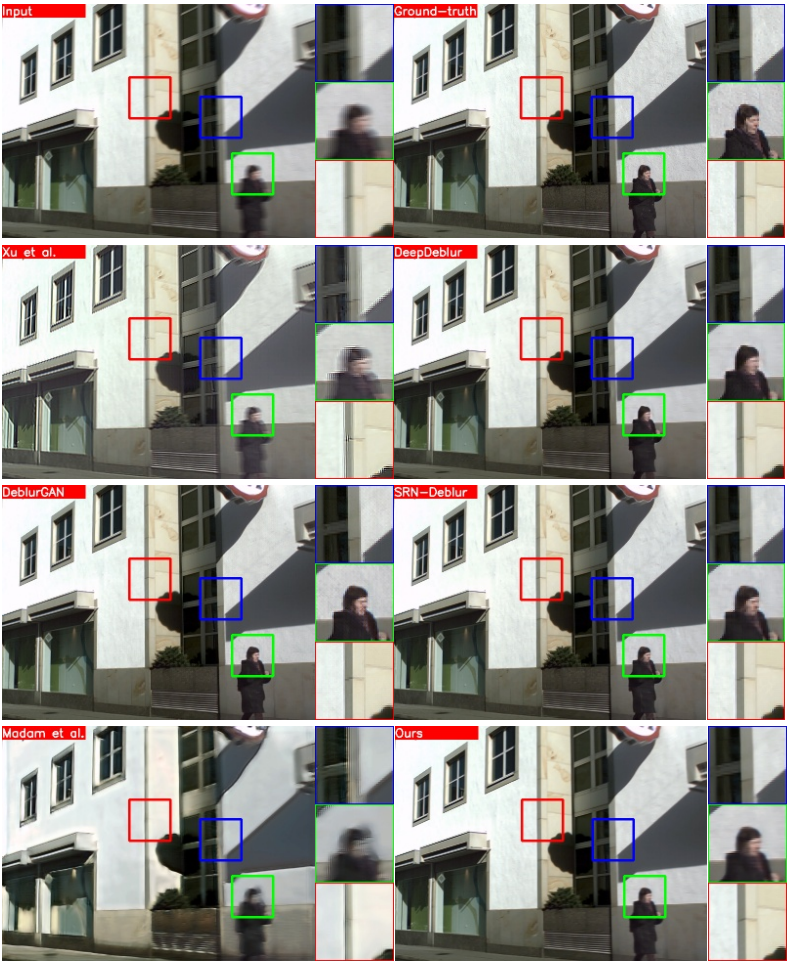


Figure 5.13.: **Qualitative comparisons on the Fastec dataset.** All the baseline networks are retrained on our Fastec dataset.

Figure 5.14.: **Temporal consistency on the Fastec dataset (frame 1-5). Left:** Ground truth. **Middle:** Blurry image. **Right:** Deblurred image by our network.

Figure 5.15.: **Temporal consistency on the Fastec dataset (frame 6-10). Left:** Ground truth. **Middle:** Blurry image. **Right:** Deblurred image by our network.

Figure 5.16.: **Temporal consistency on the Fastec dataset (frame 11-15). Left:** Ground truth. **Middle:** Blurry image. **Right:** Deblurred image by our network.

Figure 5.17.: **Temporal consistency on the Fastec dataset (frame 16-19). Left:** Ground truth. **Middle:** Blurry image. **Right:** Deblurred image by our network.

Figure 5.18.: **Temporal consistency on the real dataset. Odd rows:** Blurry image. **Even rows:** Deblurred image by our network.

Figure 5.19.: **Temporal consistency on the real dataset. Odd rows:** Blurry image. **Even rows:** Deblurred image by our network.

6. Deep Shutter Unrolling Network

6.1. Introduction

CMOS imaging sensors are widely used in many consumer products. They usually capture images with a rolling shutter mechanism. In contrast to a global shutter camera, which captures all pixels at the same time, a rolling shutter camera sequentially captures the image pixels row by row. Therefore, different types of distortions, e.g. skew, smear or wobble, will appear if the camera is moving during the image capture. It is well known that many vision tasks (e.g. structure from motion, visual odometry, pose estimation or depth prediction) suffer from rolling shutter distortions [KMR13, HFFR12, SPL16, KLR17, AKLP19, SKBP13]. The rolling shutter effect correction problem has thus received considerable attention in the past [RBR17, VMR18, ZTJ+19].

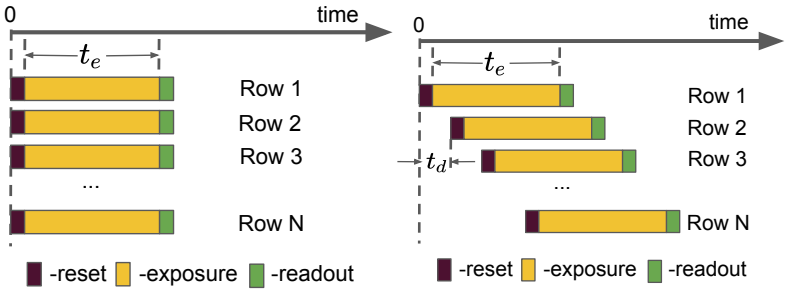
Existing works on rolling shutter effect correction can be categorized into classical approaches and single image based deep learning approaches. The classical approaches can be further categorized into single image based methods and methods which use multiple images. Single image based rolling shutter effect correction is an ill-posed problem and relies heavily on prior assumptions (e.g. straight lines must remain straight), either formulated explicitly or learned implicitly by a deep network, which limit their applicability to real scenarios. Classical multi-image based approaches are more general and instead rely on geometric constraints from multiple views to perform the rectification. However, they usually formulate it as a computationally expensive optimization problem for 6 DoF camera motions, which prevents the algorithm from being used in

Figure 6.1.: **Deep shutter unrolling network. Left:** Input rolling shutter image. **Middle:** Predicted global shutter image by our network. **Right:** Ground truth global shutter image.

time constrained applications.

Inspired by the recent success of deep neural networks on image-to-image translation problems, such as optical flow estimation [SYLK18], dense depth prediction [EPF14], motion deblurring [NKL17] and image super-resolution [LSK+17], we propose an efficient end-to-end deep neural network for rolling shutter effect correction. Our method solves a generic rectification problem from two consecutive frames. It is able to take advantage of the parallel computational power of a graphic card and runs in near real time. Furthermore, benefiting from the representational power of a deep network, our network is also able to learn good image priors to further boost the quality of the rectified image. Different from the above image-to-image translation problems, in which the estimations usually rely on its local neighborhood pixels of the input image, the rolling shutter effect correction problem is more challenging. A pixel of the rectified image might lie far away from its corresponding pixel of the input rolling shutter image, depending on the types of motion, 3D scene structure as well as its capturing time. To resolve these challenges, we propose a novel network architecture for rolling shutter image correction.

Our network takes two consecutive rolling shutter images as input and predicts the corresponding global shutter image of the latest frame. It consists of four main parts: an image encoder, a motion estimator, a differentiable forward warping block and an image decoder. The motion estimator estimates the dense per-pixel displacement field from a rolling



Global shutter Rolling shutter

Figure 6.2.: **Image formation models.** The difference between a rolling shutter camera and a global shutter camera is that the rows of a rolling shutter image are captured at different timestamps with a constant time offset t_d. For simplicity, we assume the exposure time t_e is infinitesimal throughout the paper.

shutter image to its corresponding global shutter image, given the learned feature representation from the image encoder. The differentiable forward warping block warps the learned feature representation to its corresponding global shutter representation, given the estimated displacement field. The global shutter image is then recovered by the image decoder from the warped feature representation. Our network can be trained end-to-end and only requires the ground truth global shutter image for supervision, which is easy to obtain by using a high-speed camera to synthesize the training data. Experimental results demonstrate that our method outperforms the state-of-the-art methods [ZCL17, ZTJ+19]. Fig. 6.1 presents qualitative results from our network.

Since there is no public dataset available, we also propose two novel datasets: the Fastec-RS dataset and the Carla-RS dataset, as our second contribution. The Fastec-RS dataset has 2584 image pairs. It is generated via a professional high-speed camera (with a framerate of 2400 FPS) and captured in real environments. Since the camera is mounted on a ground

vehicle which undergoes limited motion, we also create the Carla-RS dataset with general six degree of freedom (DoF) motions. This dataset is generated from a virtual 3D environment and has 2500 image pairs. To further research in this area we make both our code and the datasets public. This chapter is based on our publication from [LCLP20].

6.2. Method

The main concept of our method is to learn a dense per-pixel displacement field, which is used to warp the learned features from the rolling shutter image to its global shutter counterpart. The global shutter image is then recovered by an image decoder which decodes the warped features to an image. Our network can be trained end-to-end and only requires the global shutter image for supervision. Fig. 6.4 presents the details of our network architecture.

6.2.1. Rolling shutter image formation model

The difference between a rolling shutter camera and a global shutter camera is that every scanline of the rolling shutter camera is exposed at different timestamps, as shown in Fig. 6.2. Without loss of generality, we assume the read-out direction is from top to bottom. We further assume all pixels from the same row are captured at the same timestamp. We can thus obtain the image formation model of a rolling shutter image as follows:

$$[\mathbf{I}^r(\mathbf{x})]_i = [\mathbf{I}_i^g(\mathbf{x})]_i, \tag{6.1}$$

where $\mathbf{I}_i^g(\mathbf{x})$ is the virtual global shutter image captured at timestamp $i \cdot t_d$, t_d is the time to read out a single row, $[\mathbf{I}_i^g(\mathbf{x})]_i$ is an operator to extract the i^{th} row from an image $\mathbf{I}_i^g(\mathbf{x})$.

As the whole image readout time (i.e., Nt_d where N is the height of the image) is typically small (<50 ms), we can assume that during the time of capture the image content is primarily affected by image motion and not by other changes like object appearance or illumination. We can thus model

the virtual global shutter image $\mathbf{I}_i^g(\mathbf{x})$ as the result of the first virtual global shutter image $\mathbf{I}_0^g(\mathbf{x})$ warped by a displacement vector $\mathbf{u}_{i\to 0}$:

$$\mathbf{I}_i^g(\mathbf{x}) = \mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{i\to 0}), \tag{6.2}$$

where $\mathbf{u}_{i\to 0} \in \mathbb{R}^2$ denotes the displacement vector of pixel $\mathbf{x}$ from the i^{th} virtual global shutter image $\mathbf{I}_i^g$ to the reference image $\mathbf{I}_0^g$, which corresponds to the virtual global shutter image captured at timestamp 0. Thus, we can reformulate Eq. (6.1) to

$$[\mathbf{I}^r(\mathbf{x})]_i = [\mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{i\to 0})]_i. \tag{6.3}$$

We can further have

$$\mathbf{I}^r(\mathbf{x}) = \mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{r\to g}), \tag{6.4}$$

where $\mathbf{u}_{r\to g} \in \mathbb{R}^2$ denotes the displacement vector of pixel $\mathbf{x}$ from the rolling shutter image to the first virtual global shutter image. If we stack $\mathbf{u}_{r\to g}$ for all pixels, it has following form

$$[\mathbf{U}_{r\to g}]_i = [\mathbf{U}_{i\to 0}]_i, \tag{6.5}$$

where both $\mathbf{U}_{r\to g}$ and $\mathbf{U}_{i\to 0}$ are the dense displacement field for all pixels, in matrix form.

As a special case, if the rolling shutter camera is stationary during image capture, the displacement field $\mathbf{U}_{r\to g}$ is zero. The captured rolling shutter image equals to the global shutter image.

6.2.2. Rolling shutter effect removal

Rolling shutter effect removal is an operation to reverse the above image formation model, i.e., Eq. (6.4). In particular, it is to estimate the global shutter image $\mathbf{I}_0^g(\mathbf{x})$ given the captured rolling shutter image $\mathbf{I}^r(\mathbf{x})$. It is an ill-posed problem for single image rolling shutter effect removal, since the displacement field $\mathbf{U}_{r\to g}$ is difficult to recover from a single image. Existing works typically take advantage of prior assumptions (e.g. straight lines should remain straight) to estimate the displacement field

[RRA16,ZTJ+19]. The prior assumption can be either explicitly formulated [RRA16] or implicitly learned by a deep network [ZTJ+19]. Thus, single image rolling shutter correction methods cannot generalize to scenarios where the prior assumption is not satisfied. Therefore, we propose to use two frames to solve a more general rectification problem.

To recover $\mathbf{I}_0^g$ from $\mathbf{I}^r$, it is more convenient to have displacement field $\mathbf{U}_{g\to r}$ instead of $\mathbf{U}_{r\to g}$. The global shutter image can then be simply recovered by

$$\mathbf{I}_0^g(\mathbf{x}) = \mathbf{I}^r(\mathbf{x} + \mathbf{u}_{g\to r}), \tag{6.6}$$

where bilinear interpolation can be used for pixels which have non-integer positions. However, we are only given rolling shutter images as input. It is more difficult for us to estimate $\mathbf{U}_{g\to r}$ compared to $\mathbf{U}_{r\to g}$. Thus, we design a motion estimation network to estimate $\mathbf{U}_{r\to g}$ for rolling shutter image rectification. It is not trivial to recover the global shutter image given the displacement field $\mathbf{U}_{r\to g}$ and the rolling shutter image, since we cannot find the pixel correspondences from the global shutter image to the rolling shutter image. Thus, we propose to employ a forward warping block [FR10] to resolve this challenge. We derive and implement the derivatives of the forward warping block, to make it differentiable such that we can incorporate it into our deep network for end-to-end training. For compactness, we denote $\mathbf{I}_0^g$ as $\mathbf{I}^g$ for future sections.

6.2.3. Differentiable forward warping block

In this section, we present the detailed derivations of our differentiable forward warping block. Since we need to incorporate it as part of our network, we thus also need the partial derivatives from the outputs to the inputs. Due to the varying number of neighboring pixels, i.e., the size of $\Omega(\mathbf{x})$ from Eq. (6.7), it is not trivial to take advantage of the automatic differentiation tools from PyTorch [PGC+17] for the derivative computation. We thus derive and implement all the analytical partial derivatives by ourselves.

Forward pass: Without loss of generality, we use a particular pixel to illustrate our formulation. As shown in Fig. 6.3, we can approximate the intensity of a particular pixel from the global shutter image, as a weighted

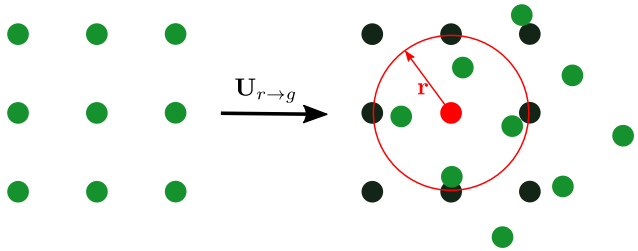


Figure 6.3.: **Differentiable forward warping.** The rolling shutter image (i.e., green pixels) is warped to the image grid of the global shutter image (i.e., black pixels and the red pixel) by the estimated displacement field $\mathbf{U}_{r\to g}$. To recover the intensities of the red pixel, we can compute the weighted average of its four neighboring pixels (i.e., the green pixels covered by the red circle with a radius r) from the rolling shutter image.

average of its neighboring pixel intensities from the rolling shutter image, which was previously used in [FR10]. Formally, this can be defined as

$$\mathbf{I}^g(\mathbf{x}) = \frac{\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} \mathbf{I}^r(\hat{\mathbf{x}})}{\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}}, \tag{6.7}$$

where $\Omega(\mathbf{x})$ is the set of all pixels $\hat{\mathbf{x}}$ from the rolling shutter image, which satisfy

$$\left\|\hat{\mathbf{x}} + \mathbf{u}_{r\to g} - \mathbf{x}\right\|_2 < r, \tag{6.8}$$

where r is a pre-defined threshold with unit in pixels. The weight is further defined as

$$\omega_{\hat{\mathbf{x}}} = e^{-\frac{d(\mathbf{x},\hat{\mathbf{x}})^2}{2\sigma^2}}, \tag{6.9}$$

where

$$d(\mathbf{x},\hat{\mathbf{x}}) = \left\|\hat{\mathbf{x}} + \mathbf{u}_{r\to g} - \mathbf{x}\right\|_2, \tag{6.10}$$

and σ is a pre-defined width of the kernel function.

Backward pass: From the above equations, we can find the inputs of the formation model are the rolling shutter image $\mathbf{I}^r(\hat{\mathbf{x}})$ and the optical flow $\mathbf{u}_{r\to g}$. The output is the global shutter image $\mathbf{I}^g(\mathbf{x})$. In order to incorporate it into our network, we thus need to have both $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{I}^r(\hat{\mathbf{x}})}$ and $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{u}_{r\to g}}$. Without loss of generality, we take a particular channel of a particular pixel to derive the partial derivatives as follows.

$$\frac{\partial I_c^g(\mathbf{x})}{\partial I_c^r(\mathbf{x}')} = \frac{\omega_{\mathbf{x}'}}{\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})}\omega_{\hat{\mathbf{x}}}}, \tag{6.11}$$

where $I_c^g(\mathbf{x})$ and $I_c^r(\mathbf{x}')$ are the c^{th} channel intensity of pixel $\mathbf{x}$ and $\mathbf{x}'$ respectively, $\mathbf{x}'$ is an element from $\Omega(\mathbf{x})$. Similarly, we can have

$$\frac{\partial I_c^g(\mathbf{x})}{\partial \mathbf{u}_{r\to g}} = \frac{\partial I_c^g(\mathbf{x})}{\partial \omega_{\mathbf{x}'}}\frac{\partial \omega_{\mathbf{x}'}}{\partial \mathbf{u}_{r\to g}}, \tag{6.12}$$

where $\mathbf{u}_{r\to g}$ is the displacement vector of pixel $\mathbf{x}'$. From Eq. (6.7), we can further get

$$\frac{\partial I_c^g(\mathbf{x})}{\partial \omega_{\mathbf{x}'}} = \frac{I_c^r(\mathbf{x}')\cdot\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})}\omega_{\hat{\mathbf{x}}} - \sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})}\omega_{\hat{\mathbf{x}}} I_c^r(\hat{\mathbf{x}})}{(\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})}\omega_{\hat{\mathbf{x}}})^2}. \tag{6.13}$$

Similarly, from Eq. (6.9), we can get

$$\frac{\partial \omega_{\mathbf{x}'}}{\partial \mathbf{u}_{r\to g}} = \frac{\partial \omega_{\mathbf{x}'}}{\partial d^2}\frac{\partial d^2}{\partial \mathbf{u}_{r\to g}}, \tag{6.14}$$

where

$$d^2 = \|\mathbf{x}' + \mathbf{u}_{r\to g} - \mathbf{x}\|_2^2 = (x' + u_x - x)^2 + (y' + u_y - y)^2, \tag{6.15}$$

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{6.16}$$

$$\mathbf{u}_{r\to g} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \tag{6.17}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{6.18}$$

We can further get

$$\frac{\partial \omega_{\mathbf{x}'}}{\partial d^2} = -\frac{1}{2\sigma^2} e^{-\frac{d^2}{2\sigma^2}}, \tag{6.19}$$

$$\frac{\partial d^2}{\partial \mathbf{u}_{r\to g}} = \begin{bmatrix} \frac{\partial d^2}{\partial u_x} & \frac{\partial d^2}{\partial u_y} \end{bmatrix} = \begin{bmatrix} 2(x' + u_x - x) & 2(y' + u_y - y) \end{bmatrix}. \tag{6.20}$$

Implementation details: From the above equations, we can find that we need to get $\Omega(\mathbf{x})$ before we compute both the forward pass and the backward pass. However, a direct implementation of the above equations is in-efficient, due to the varying size of $\Omega(\mathbf{x})$. We therefore propose an efficient implementation to solve the above challenge, such that it can fully take advantage of the computational capability of a graphic card.

Instead of pre-computing $\Omega(\mathbf{x})$ by checking every pixel of the global shutter image, we iterate over every pixel of the rolling shutter image. For each pixel $\mathbf{x}'$ of the rolling shutter image, we accumulate all the necessary computations related to pixel $\mathbf{x}$ from the global shutter image, which satisfies

$$\|\mathbf{x}' + \mathbf{u}_{r\to g} - \mathbf{x}\|_2 < r, \tag{6.21}$$

where r is the pre-defined threshold. Furtheremore, we can also re-use the already accumulated $\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}$ and $\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} \mathbf{I}^r(\hat{\mathbf{x}})$ from the forward pass, for the backward pass. By following this approach, all the pixels $\mathbf{x}'$ can be processed in parallel, which makes the implementation very efficient on a graphic card.

6.2.4. Network architecture

In this section, we explain how to design a deep network to estimate $\mathbf{U}_{r\to g}$ and recover the global shutter image. Everything presented in the previous sections can be directly generalized from pixels to learned feature representations. Fig. 6.4 presents the architecture of our network.

Our network accepts two consecutive rolling shutter images and outputs a global shutter image corresponding to the latest frame. It consists of

four main parts, i.e., an encoder network, a motion estimation network, a differentiable forward warping block and an image decoder network. The encoder network consists of three pyramid levels. Each level has a convolutional layer followed by three residual blocks. To recover the latent global shutter image, it would be easier for the image decoder network to operate on the feature representation which corresponds to the global shutter image. We therefore use the differentiable forward warping block to transform the learned feature representation of the latest rolling shutter image to its global shutter counterpart. The displacement field $\mathbf{U}_{r \to g}$ used by the forward warping block is estimated by the motion estimation network.

Besides the camera motion and 3D scene geometry, $\mathbf{U}_{r \to g}$ also depends on the time when a particular pixel is being captured, i.e., the displacement vector of a pixel nearer to the first row is usually smaller than those are further away. During our ablation study, we find that the motion estimation network has difficulty to learn/model this implicitly. We thus model this dependency explicitly and design the network to learn the dense velocity field instead, as shown in Fig. 6.4. Our motion estimation network computes the cost volumes between both frames by a correlation layer [IMS+17], based on the learned feature representation. The velocity field is then estimated by a dense network block, given the computed cost volumes as input. To recover the displacement field $\mathbf{U}_{r \to g}$, we multiply the estimated velocity field with the time offset (i.e., $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ as shown in Fig. 6.4) between the captured pixel and that of the first row. $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ represent the time-offset for different pyramid levels and they have the same resolutions as the feature representations of the corresponding pyramid levels. Without calibrating the camera, we simply set the row read-out time (i.e., t_d as shown in Fig. 6.2) as 1 for simplicity. The image decoder then predicts the global shutter image given the warped feature representation. The decoder network also consists of three pyramid levels. Each level has three residual blocks followed by a deconvolution layer. We present the details of our network architecture and all the other baseline networks which will be evaluated later in our ablation study as follows.

Deep shutter unrolling network: Fig. 6.4 demonstrates the overall archi-

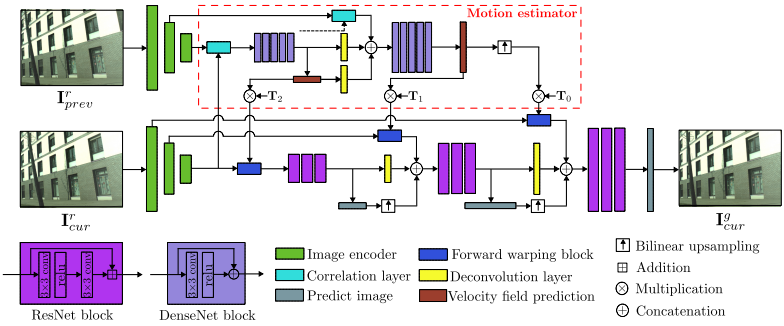


Figure 6.4.: **Overall architecture of the deep shutter unrolling network.**

tecture of our deep shutter unrolling network. It consists of four main parts: an image encoder network, a motion estimation network, a differentiable forward warping block and an image decoder network. The details of the image encoder network are shown in Fig. 6.5. It consists of three pyramid levels. Each level has a convolutional layer followed by a ReLU activation function and three ResNet blocks [HZRS16]. The convolutional layer of the first pyramid level has a kernel size 7×7, 32 filters and uses a stride size 1. The convolutional layers of both the second pyramid level and the third pyramid level have a kernel size 3×3 and use a stride size 2 to down-sample the learned feature representations. The number of filters are 64 and 128 respectively. The number of filters for the corresponding ResNet blocks are 32, 64 and 128 for the first, the second and the third pyramid levels, respectively.

The motion estimation network estimates the dense displacement field $\mathbf{U}_{r \to g}$ from the rolling shutter image to its corresponding global shutter image. It computes the matching cost volumes based on the learned feature representations by the image encoder. The matching cost volumes are computed by a correlation layer, which is usually used in optical flow prediction networks [SYLK18]. Five DenseNet blocks [HLW17] are used to learn the dense velocity field for both the third pyramid level and the

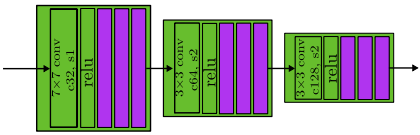


Figure 6.5.: **Image encoder network.**

second pyramid level, from the computed matching cost volumes. We bi-linearly upsample the estimated dense velocity field from the second pyramid level for the first pyramid level as shown in Fig. 6.4. The DenseNet block consists of a convolutional layer followed by a ReLU activation function. The convolutional layer has a 3×3 kernel size and a stride size 1. Details can be found in Fig. 6.4. The number of filters for the five DenseNet blocks are 128, 128, 96, 64 and 32 for the third pyramid level. Similarly, we have 64, 64, 48, 32 and 16 for the second pyramid level. The deconvolutional layers have a kernel size 4×4, a stride size 2, a padding size 1 and 2 filters. The velocity field prediction layer has a 2D convolutional layer with a kernel size 3×3, a stride size 1, a padding size 1, and 2 filters.

The image decoder network has three pyramid levels. Each pyramid level has three ResNet blocks [HZRS16] followed by a deconvolutional layer and an image prediction layer. All the ResNet blocks have two convolutional layers with a kernel size 3×3, a stride size 1 and a padding size 1. The number of filters is equal to the input number of channels. In particular, the number of filters are 128, 64+3+3 and 32+3+3 respectively. The deconvolutional layers have a kernel size 4×4, a stride size 2, a padding size 1 and 3 filters. The image prediction layer has a 2D convolutional layer with a kernel size 3×3, a stride size 1, a padding size 1, and 3 filters.

Net-autoenc networks: Fig. 6.6 demonstrates the architecture of the Net-autoenc-1 network. It shares the same image encoder and decoder as our deep shutter unrolling network. The Net-autoenc-2 network is obtained by modifying the first convolutional layer of the image encoder network to accept two concatenated images as input. The rest layers are the same as

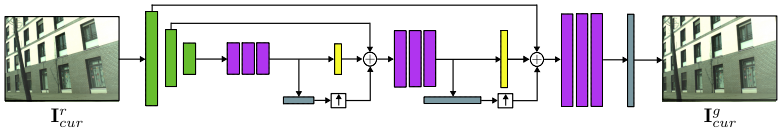


Figure 6.6.: **Architecture of Net-autoenc-1 network.**

that of the Net-autoenc-1 network.

Net-disp network: The Net-disp network has the same architecture as the deep shutter unrolling network. We simply set the time matrices to be $\mathbf{1}$, i.e., $\mathbf{T}_0 = \mathbf{1}$, $\mathbf{T}_1 = \mathbf{1}$ and $\mathbf{T}_2 = \mathbf{1}$.

6.2.5. Loss functions

To train our network, only the corresponding ground truth global shutter image $\mathbf{I}^g_{gt}$ is required. Empirically, we find a linear combination of the pixel-wise $\mathcal{L}_1$ loss, the perceptual loss $\mathcal{L}_p$ [JAL16] and a total variation loss $\mathcal{L}_{tv}$ to encourage piecewise smoothness in the estimated displacement field $\mathbf{U}_{r\to g}$, can give satisfactory performance. If the perceptual loss is omitted, the estimated image tends to be blurry. In summary, our loss function can be formulated as

$$\mathcal{L} = \mathcal{L}_p(\mathbf{I}^g, \mathbf{I}^g_{gt}) + \lambda_1 \mathcal{L}_1(\mathbf{I}^g, \mathbf{I}^g_{gt}) + \lambda_2 \mathcal{L}_{tv}(\mathbf{U}_{r\to g}), \tag{6.22}$$

where both λ_1 and λ_2 are hyper-parameters and determined empirically, $\mathbf{I}^g$ is the estimated global shutter image and $\mathbf{I}^g_{gt}$ is the ground truth global shutter image.

6.3. Datasets

Both Rengarajan et al. [RBR17] and Zhuang et al. [ZTJ+19] propose individual datasets to train their networks respectively. However, their datasets are not released to the community. Furthermore, both datasets simplify

the real formation process of a rolling shutter image. For example, Rengarajan et al. [RBR17] generate the synthetic image by applying simple affine image warping and does not consider the 3D geometry. Zhuang et al. [ZTJ+19] do consider the effect of the 3D geometry. They warp a single global shutter image from the KITTI dataset [GLU12] to generate a synthetic rolling shutter image, given the corresponding dense depth map and camera motion. The dense depth map is estimated from a stereo camera by a depth prediction network [CC18]. The 6 DoF camera motion is randomly sampled from a pre-defined interval. However, it still simplifies the real image formation process, e.g. their dataset does not model occlusions, which are common in real-world scenarios. Furthermore, the estimated dense depth map is not the true 3D scene geometry either.

Thus, we propose two datasets: the Carla-RS dataset and the Fastec-RS dataset. Our datasets are synthesized via high speed cameras and simulate the real image formation process. The Carla-RS dataset is generated from a virtual 3D environment provided by the Carla simulator [DRC+17]. Carla simulator is an open-source platform for autonomous driving research and it provides seven photorealistic 3D virtual towns. We implement a rolling shutter camera model since the original simulator does not support it. We also relax the constraint that the camera is mounted on a ground vehicle, such that we can freely move our rolling shutter camera in six DoF. 250 sequences are randomly sampled and each sequence has 10 consecutive frames. Both a constant translational velocity model and a constant angular rate model are used for the sequence generation, which is typically hold in real scenarios due to the short time interval (i.e., $<$50ms) between two consecutive frames. In total, we generate 2500 rolling shutter images at a resolution of 640×448 pixels.

Since the Carla-RS dataset is generated from a virtual environment, we also propose another dataset, the Fastec-RS dataset which is created using real images in the wild. The Fastec-RS dataset is synthesized using a professional Fastec TS5[1] high speed global shutter camera with a framerate of 2400 FPS. We mount the camera on a ground vehicle and collect 76 image sequences at a resolution of 640×480 pixels in mainly urban

[1] https://www.fastecimaging.com/fastec-high-speed-cameras-ts-series/

Figure 6.7.: **Qualitative comparisons against state-of-the-art methods on the Carla-RS dataset. Second row:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}_{gt}^{g}$.

environment. Each sequence synthesizes 34 rolling shutter images. In total, we have 2584 image pairs. The rolling shutter image is synthesized by sequentially copying a row of pixels from the captured global shutter images.

6.4. Experimental evaluation

Datasets: We evaluate our algorithm with both the Carla-RS dataset and Fastec-RS dataset. We split the Carla-RS dataset into training data and test data. The training data has 210 sequences and the test data has 40 sequences. Similarly, we split the Fastec-RS dataset into 56 sequences for training and 20 sequences for test. Both the training data and test data have no overlapping scenes. Since the ground truth occlusion masks can be obtained and are also provided by the Carla-RS dataset, we thus compute two quantitative metrics for better evaluation, i.e., one without using the occlusion mask and the other one using the occlusion mask. For compactness, we denote the Carla-RS dataset with masks, the Carla-RS

Figure 6.8.: **Qualitative comparisons against conventional methods with dataset of [ZCL17].** It demonstrates that our network predicts a plausible rectification and also inpaints the occluded regions with the learned image priors.

dataset without masks and the Fastec-RS dataset as **CRM**, **CR** and **FR** respectively for quantitative evaluations.

Implementation details: We implemented our network by using PyTorch [PGC+17]. The differentiable forward warping is imeplemted in CUDA with PyTorch wrappers. The hyper-parameters are set empirically to $r = 2$, $\sigma = 0.5$, $\lambda_1 = 10$ and $\lambda_2 = 0.1$ unless stated otherwise. For better convergence, we train our network in three pyramid levels. The network is trained in 200 epochs with a learning rate 10^{-4}. We use a batch size of 3 and use uniform random crop at a resolution of 320×256 pixels for data augmentation.

State-of-the-art methods: We compare our network against two state-of-the-art methods from [ZCL17] and [ZTJ+19], which are the two most related works to our approach. The method from [ZCL17] is a classical two image based approach and we use the implementation provided by the authors. The method from [ZTJ+19] is a single image based deep learning approach. Since the authors did not release their implementations, we reimplemented their network and trained it on our datasets for fair comparisons. To ensure our implementation is correct, we generated the same dataset

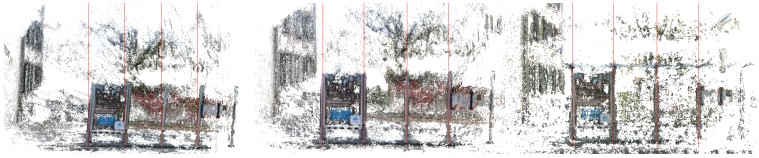


Figure 6.9.: **Generalization performance on real data. Left:** Reconstructed 3D model with input rolling shutter images. **Middle:** Reconstructed 3D model with predicted global shutter images. **Right:** Reconstructed 3D model with real global shutter images.

	PSNR↑ (dB)			SSIM↑	
Networks	CRM	CR	FR	CR	FR
Net-autoenc-1	19.04	18.96	23.41	0.60	0.70
Net-autoenc-2	21.39	21.33	26.07	0.67	0.75
Net-disp	21.24	21.10	25.74	0.67	0.73
Net-vel-self	27.31	26.87	26.77	0.82	0.76
Ours	**27.78**	**27.30**	**27.04**	**0.84**	**0.77**

Table 6.1.: **Ablation study on the network architectures and loss function.**

as described by [ZTJ+19] and trained our implemented network with it. In our experiments, the test performance is similar to what was reported in [ZTJ+19], in terms of both quantitative and qualitative metrics on their dataset. Since the dataset from [ZTJ+19] is for single image based method, we cannot evaluate our algorithm with that dataset.

Evaluation metrics: We use the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) for quantitative comparisons. Both PSNR and SSIM metrics are commonly used to measure the similarity between images (see e.g. [NKL17, LSK+17]). Larger PSNR/SSIM values indicate better image quality.

Ablation study on the network architectures: We implemented several

baseline networks to justify the design of our network architecture. We remove the motion estimation network and differentiable forward warping block to have a vanilla auto-encoder network. We also modify the image encoder such that it can accept a single rolling shutter image as input. In total, we have two auto-encoder networks for comparison. We denote them with **Net-autoenc-1** and **Net-autoenc-2** respectively. We also study the performance if we learn the displacement field directly, instead of the velocity field as described in Section 6.2.4. It is achieved by setting $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ all equal to $\mathbf{1}$, such that the estimated velocity field equals to the displacement field. We denote the network as **Net-disp**.

Table 6.1 presents the quantitative performances of the networks. It demonstrates that a vanilla auto-encoder network has difficulties to learn a good representation for rolling shutter effect removal. A possible reason is that the rectification problem involves non-local operations, which challenge the representation power of a vanilla auto-encoder network. Besides the dependencies of $\mathbf{U}_{r\to g}$ on the camera motion and 3D scene geometry, it also depends on the capture time of a particular pixel. We find it challenges the motion estimation network to estimate the displacement field directly. It is well explained by our experimental results, i.e., our network outperforms **Net-disp**. Furthermore, the experimental results also demonstrate that **Net-autoenc-2** network performs better than **Net-autoenc-1** network with around 2.37 dB improvement on the Carla-RS dataset and 2.66 dB improvement on the Fastec-RS dataset. It demonstrates that it is more difficult to learn a good representation with a single rolling shutter image compared to multiple images, due to the ill-posed nature of single image based method.

Ablation study on the loss function: We did an ablation study to justify the loss functions that we used for network training, i.e., Eq. (6.22). We focus our attention on the explicitly supervision of the dense displacement field estimation. To achieve this, we introduce an additional loss function $\mathcal{L}_d$

$$\mathcal{L}_d = \left\| \mathbf{I}^r - \mathcal{W}_{g\to r} \circ \mathbf{I}^g_{gt} \right\|_1 , \tag{6.23}$$

where $\mathbf{I}^r$ is the latest input rolling shutter image, $\mathbf{I}^g_{gt}$ is the corresponding ground truth global shutter image, $\mathcal{W}_{g\to r}$ is an operator which warpes

the global shutter image to its corresponding rolling shutter image and it depends on the estimated dense displacement field $\mathbf{U}_{r\to g}$. The warping is achieved by bilinear interpolations. The final loss function used to train the network can be formualted as

$$\mathcal{L}_f = \mathcal{L} + \lambda_1 \mathcal{L}_d, \tag{6.24}$$

where $\mathcal{L}$ represents the original loss function as shown in Eq. (6.22), λ_1 is a hyper-parameter and is empiracally selected as 10. We represent the network trained with the loss function $\mathcal{L}_f$ as **Net-vel-self**.

We train **Net-vel-self** with the same parameter configurations as other networks. Experimental results presented in Table 6.1 demonstrate that our network which is trained with only $\mathcal{L}$ performs better than **Net-vel-self**. The introduction of the self-supervision loss $\mathcal{L}_d$ for the dense displacement field $\mathbf{U}_{r\to g}$ does not help improve the performance of global shutter image estimation. A possible explanation is that the occlusions between $\mathbf{I}^r$ and $\mathbf{I}^g_{gt}$, which are used to supervise the learning of $\mathbf{U}_{r\to g}$, degrades the prediction of $\mathbf{U}_{r\to g}$ since we cannot do bi-directional occlusion detection. The forward feature warping for global shutter image recovery would thus be affected by the degraded $\mathbf{U}_{r\to g}$ and it further affects the final global shutter image prediction. It demonstrates that the loss function $\mathcal{L}$ presented in Eq. (6.22) is sufficient to implicitly supervise the learning of $\mathbf{U}_{r\to g}$.

Quantitative and qualitative evaluations against baseline methods: We compare our network with two state-of-the-art baseline methods. Both the quantitative and qualitative comparisons are presented in Table 6.2 and Fig. 6.7 respectively. The experimental results demonstrate that our method performs better than the other two state-of-the-art approaches. The work from Zhuang et al. [ZTJ+19] is a single image learning based approach. We find that it has limited generalization performance on our test data. The reason is that the scene content of our test data is quite different from that of our training data. The learned geometric priors from training data do not hold to the test data. In contrast, our network can be generalized well as we solve a generic rectification problem with two input frames. Zhuang et al. [ZCL17] is a classical approach with two input frames. We find it can work well if the input images have good textures. However, as shown

	PSNR↑ (dB)			SSIM↑	
Methods	CRM	CR	FR	CR	FR
Zhuang et al. [ZCL17]	25.93	22.88	21.44	0.77	0.71
Zhuang et al. [ZTJ+19]	18.70	18.47	N.A.	0.58	N.A.
Ours	**27.78**	**27.30**	**27.04**	**0.84**	**0.77**

Table 6.2.: **Quantitative comparisons against the state-of-the-art methods.** Since the Fastec-RS dataset does not have ground truth depth and motion, we cannot evaluate Zhuang et al. [ZTJ+19] with it.

in Fig. 6.7, it does not perform well for input frames with poorly textured regions, which results in visually unpleasing global shutter images. In contrast, our network predicts a plausible rectification with better image quality. Furthermore, our network can also take advantage of the learned image priors to fill in the occluded regions, from which the classical approach (i.e., Zhuang et al. [ZCL17]) is unable to reconstruct. This is also visible in the quantitative results, shown in Table 6.2, i.e., the PSNR metrics for Zhuang et al. [ZCL17] on Carla-RS dataset are 25.93 dB and 22.88 dB for the evaluations with occlusion masks and without masks respectively; The difference is 3.05 dB, however, ours is only 0.48 dB. It demonstrates our method handles occlusion better, since our network is able to inpaint the occluded regions from the learned image priors. Furthermore, our network is also orders of magnitude faster than Zhuang et al. [ZCL17]. It takes around 0.43 second to process a VGA resolution image (i.e., 640×480 pixels) with an Nvidia GTX 1080Ti graphic card, while Zhuang et al. [ZCL17] takes around 467.26 seconds on an Intel Core i7-7700K CPU. More qualitative results can be found from our supplementary material.

Generalization performance to real data: To evaluate the generalization performance of our network, we also collect a sequence of real rolling shutter images with a Logitech C210 webcam. The camera is mounted on the side of a ground vehicle which moves forward. For comparison, we also collect global shutter images with the same camera (i.e., the camera is

stationary while capture). The rolling shutter images are then rectified with our pretrained network. We run a SfM pipeline (i.e., COLMAP [SF16]) to process the rolling shutter images, the rectified rolling shutter images, and the global shutter images respectively. Fig. 6.9 demonstrates that our pretrained network corrects the distortion and results in a more accurate 3D model as the ground truth model. We also evaluate the generalization performance of our network against conventional methods with the dataset from Zhuang et al. [ZCL17]. The results presented in Fig. 6.8 demonstrate that our network predicts a plausible rectification and also inpaints the occluded regions with the learned image priors.

Additional qualitative experimental results: In this section, we present additional experimental results. Both Fig. 6.10 and Fig. 6.11 demonstrate the qualitative comparisons against the baseline methods on the Carla-RS and Fastec-RS datasets, respectively. We also evaluate the temporal consistency of our network. Fig. 6.12 presents the predicted global shutter images from a continuous image sequence. The image sequence is captured by a real rolling shutter camera and used to reconstruct the 3D model shown in Fig. 6.9. The experimental results demonstrate that our network is able to estimate temporally consistent global shutter images.

6.5. Conclusion

We propose an efficient end-to-end deep neural network for generic rolling shutter image correction. Our network takes two consecutive frames and estimates the global shutter image corresponding to the latest frame. It is able to take advantage of the representational power of a deep network and outperforms existing state-of-the-art methods. We also present two large datasets, which simulate the real image formation process of a rolling shutter image.

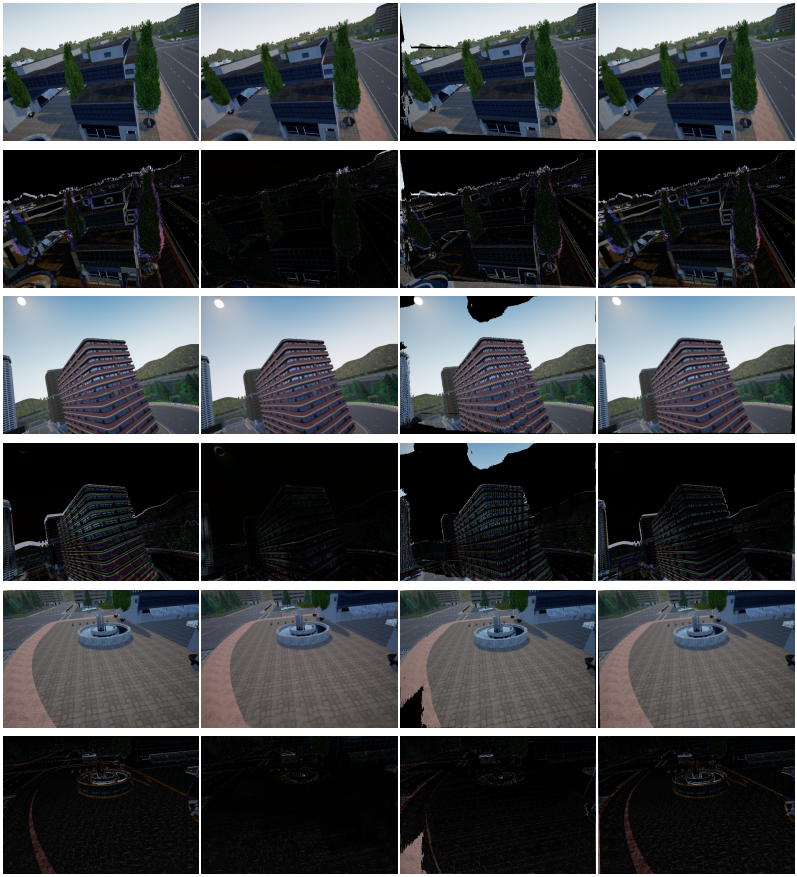


Figure 6.10.: **Qualitative comparisons against baseline methods with the Carla-RS dataset. Even rows:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}^g_{gt}$. **First column:** Input rolling shutter image. **Second column:** Predicted global shutter image by our network. **Third column:** Predicted global shutter image by Zhuang et al. [ZCL17]. **Fourth column:** Predicted global shutter image by Zhuang et al. [ZTJ+19].

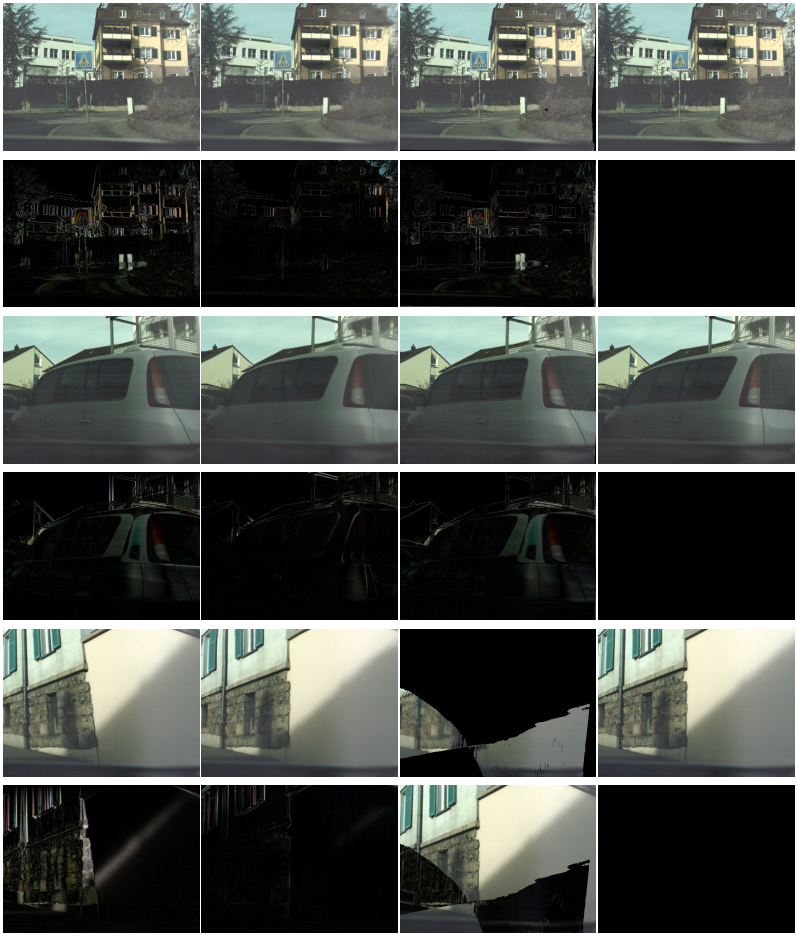


Figure 6.11.: **Qualitative comparisons against baseline methods with the Fastec-RS dataset. Even rows:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}_{gt}^{g}$. **First column:** Input rolling shutter image. **Second column:** Predicted global shutter image by our network. **Third column:** Predicted global shutter image by Zhuang et al. [ZCL17]. **Fourth column:** Ground truth global shutter image.

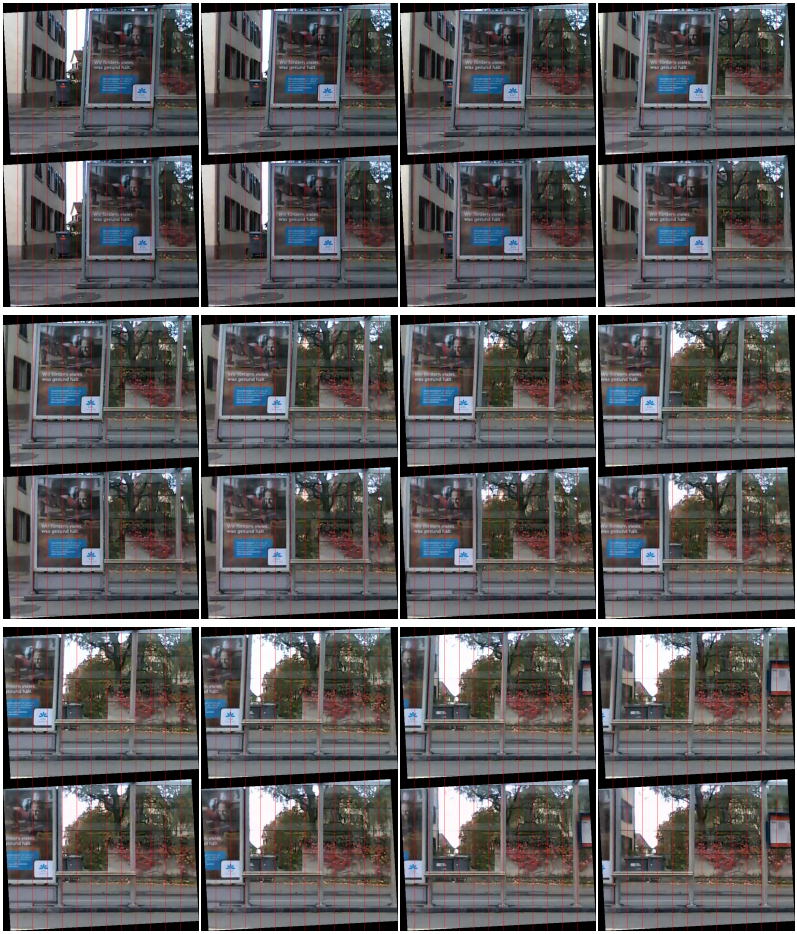


Figure 6.12.: **Qualitative evaluation with a real rolling shutter image sequence. Odd rows:** Input rolling shutter image. **Even rows:** Predicted global shutter image by our pretrained model. For better visualization, the images are rotated by a constant offset to compensate the misalignment between the camera and the gravity direction. It demonstrates that our network is able to estimate temporally consistent global shutter images.

Part III.

Algorithmic Perspective

7. Motion Blur Aware Robust Visual Odometry

7.1. Introduction

While many algorithms have been proposed, motion blur is still a major challenge remaining for visual odometry methods. Motion blur is one of the most common artifacts that degrade images. It usually occurs in low-light conditions where longer exposure times are necessary. This affects both feature based approaches (e.g. ORB-SLAM [MAT17a]), which struggle to detect keypoints, and direct methods (e.g. DSO [EKC17]) which rely on strong image gradients for their alignment. While relocalization strategies can partially mitigate the problem by allowing the VO to recover after losing track, VO would still fail if the camera continues to move in un-explored areas.

In this chapter, we propose a novel hybrid visual odometry method which is robust to motion blur, which is based on our publication from [LZVP21]. As conventional algorithms, our method consists of a front-end tracker and a back-end mapper. During tracking, instead of estimating the camera pose at a particular point in time, we estimate the local camera motion trajectory within the exposure time for each frame. This allows us to explicitly model the motion blur in the image and leverage it for tracking. We assume that the reference keyframe image is sharp, which is achieved by applying a deep deblurring network on the original motion blurred image. Since keyframes are usually sampled with a frequency much lower than frame-rate (and are less sensitive to latency), we can thus take advantage of a powerful deep network for keyframe deblurring (e.g. [TGS+18]). To estimate the camera motion trajectory during image capture, we locally re-

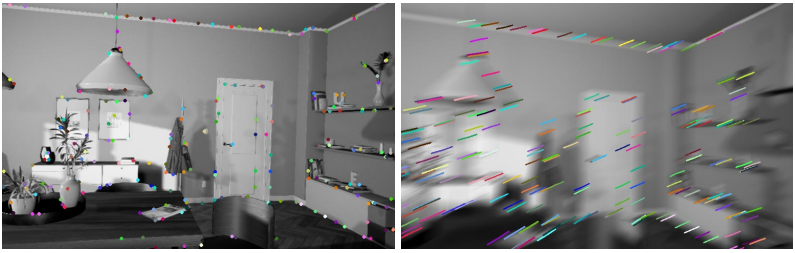


Figure 7.1.: *Motion Blur Aware Visual Odometry*. We propose a full pipeline for performing motion blur aware visual odometry. By explicitly modelling the image formation process during tracking, we can actively compensate for motion blur in the direct image alignment.

blur the sharp reference keyframe image that is then used for direct image alignment against current tracked frame. The back-end jointly optimizes the camera poses and scene geometry based on the deblurred keyframe images, by maximizing the photometric consistency. We build our method on the popular DSO [EKC17] framework. As another contribution, we also propose a novel benchmarking dataset targeting motion blur aware VO. Our dataset contains sequences with varying levels of motion blur. Time synchronized ground truth trajectories are also provided by an accurate indoor motion capturing system. By making this dataset publicly available to the community, we hope to encourage further research on making VO robust, which is important for real-world deployments.

We evaluate our approach with both synthetic dataset and real datasets. The experimental results demonstrate that we are able to improve the robustness of the visual odometry, while keeping comparable accuracy as that for images without motion blur. Furthermore, our motion blur aware VO (called *MBA-VO*) is also able to run in real-time on a laptop with a Nvidia GeForce RTX 2080 graphic card.

7.2. Preliminaries

For the ease of illustration, we define following notations. We denote scalar with lower case letter (e.g. λ); we denote vector with bold lower case letter (e.g. $\mathbf{x}$); we denote matrix with bold upper case letter (e.g. $\mathbf{T}$); we denote a point $\mathbf{p}$ in coordinate frame $\mathcal{F}_a$ with $\mathbf{p}^a$; the transformation matrix $\mathbf{T}_a^b \in \mathbf{SE}(3)$ transforms a 3D point $\mathbf{p}^a$ in coordinate frame $\mathcal{F}_a$ to coordinate frame $\mathcal{F}_b$; we further decompose $\mathbf{T}_a^b$ with $\mathbf{R}_a^b \in \mathbf{SO}(3)$ and $\mathbf{t}_a^b \in \mathbb{R}^3$, such that

$$\mathbf{p}^b = \mathbf{T}_a^b \cdot \mathbf{p}^a = \mathbf{R}_a^b \cdot \mathbf{p}^a + \mathbf{t}_a^b, \tag{7.1}$$

where $\mathbf{t}_a^b$ and $\mathbf{R}_a^b \in \mathbf{SO}(3)$ represents the translation and orientation respectively; we use unit quaternion to represent the orientation $\mathbf{R}_a^b$, i.e. $\bar{\mathbf{q}}_a^b = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$, where T represents the transpose operator.

Quaternion multiplication: Given two unit quaternions,

$$\bar{\mathbf{q}}_0 = \begin{bmatrix} q_{x0} \\ q_{y0} \\ q_{z0} \\ q_{w0} \end{bmatrix}, \quad \bar{\mathbf{q}}_1 = \begin{bmatrix} q_{x1} \\ q_{y1} \\ q_{z1} \\ q_{w1} \end{bmatrix}, \tag{7.2}$$

we can compute their product as

$$\bar{\mathbf{q}} = \bar{\mathbf{q}}_0 \otimes \bar{\mathbf{q}}_1 = \mathbf{Q}(\bar{\mathbf{q}}_0) \cdot \bar{\mathbf{q}}_1 = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_1) \cdot \bar{\mathbf{q}}_0, \tag{7.3}$$

where $\otimes$ is the product operator for quaternions, and $\mathbf{Q}(\bar{\mathbf{q}}_0)$ and $\hat{\mathbf{Q}}(\bar{\mathbf{q}}_1)$ can be defined as

$$\mathbf{Q}(\bar{\mathbf{q}}_0) = \begin{bmatrix} q_{w0} & -q_{z0} & q_{y0} & q_{x0} \\ q_{z0} & q_{w0} & -q_{x0} & q_{y0} \\ -q_{y0} & q_{x0} & q_{w0} & q_{z0} \\ -q_{x0} & -q_{y0} & -q_{z0} & q_{w0} \end{bmatrix}, \tag{7.4}$$

$$\hat{\mathbf{Q}}(\bar{\mathbf{q}}_1) = \begin{bmatrix} q_{w1} & q_{z1} & -q_{y1} & q_{x1} \\ -q_{z1} & q_{w1} & q_{x1} & q_{y1} \\ q_{y1} & -q_{x1} & q_{w1} & q_{z1} \\ -q_{x1} & -q_{y1} & -q_{z1} & q_{w1} \end{bmatrix}. \tag{7.5}$$

The Jacobian can be derived as

$$\frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}_0} = \bar{\mathbf{Q}}(\bar{\mathbf{q}}_1), \qquad \frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}_1} = \mathbf{Q}(\bar{\mathbf{q}}_0). \tag{7.6}$$

Inverse of unit quaternion: Given a unit quaternion,

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}, \tag{7.7}$$

its inverse is simply the conjugate $\bar{\mathbf{q}}^{-1}$ and can be defined as

$$\bar{\mathbf{q}}^{-1} = \begin{bmatrix} -q_x \\ -q_y \\ -q_z \\ q_w \end{bmatrix}. \tag{7.8}$$

The Jacobian can be derived as

$$\frac{\partial \bar{\mathbf{q}}^{-1}}{\partial \bar{\mathbf{q}}} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.9}$$

7.3. Method

In this section we present our motion blur aware visual odometry. We build on Direct Sparse Odometry (DSO) from Engel et al. [EKC17]. The proposed pipeline consists of three main parts: a motion blur aware visual tracker, a keyframe deblurring network and a local mapper.

The front-end tracker estimates the camera motion trajectory within the exposure time of current blurry frame, relative to the latest sharp keyframe image. Each new keyframe is processed with the motion deblurring network. The local mapper then jointly optimizes the camera poses and the scene

structure based on the recovered latent sharp keyframe images. We use the same local mapper as in DSO [EKC17] and the main technical contribution of our work is the motion blur aware tracker, which we will detail in the following sections.

7.3.1. Motion blur image formation model

The physical image formation process of a digital camera, is to collect photons during the exposure time and convert them into measureable electric charges. This process can be mathematically modelled as integrating over a set of virtual sharp images:

$$\mathbf{B}(\mathbf{x}) = \lambda \int_0^\tau \mathbf{I}_{\mathrm{t}}(\mathbf{x})\mathrm{dt}, \tag{7.10}$$

where $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{\mathrm{W}\times\mathrm{H}\times 3}$ is the captured image, W and H are the width and height of the image respectively, $\mathbf{x} \in \mathbb{R}^2$ represents the pixel location, λ is a normalization factor, τ is the camera exposure time, $\mathbf{I}_{\mathrm{t}}(\mathbf{x}) \in \mathbb{R}^{\mathrm{W}\times\mathrm{H}\times 3}$ is the virtual sharp image captured at timestamp t within the exposure time. Motion in the camera during the exposure time will result in different virtual images $\mathbf{I}_{\mathrm{t}}(\mathbf{x})$ for each t, resulting in a blurred image $\mathbf{B}(\mathbf{x})$. The model can be discretely approximated as

$$\mathbf{B}(\mathbf{x}) \approx \frac{1}{n}\sum_{i=0}^{n-1} \mathbf{I}_{\mathrm{i}}(\mathbf{x}), \tag{7.11}$$

where n is the number of discrete samples.

The amount of motion blur in an image thus depends on the motion during the exposure time. For shorter exposure time, the relative motion will be small even for a quickly moving camera. Conversely, for long exposure time (e.g. in low light conditions), even a slowly moving camera can result in a motion blurred image.

7.3.2. Direct image alignment with sharp images

Before introducing our direct image alignment algorithm with blurry images, we first review the original algorithm with sharp images. Direct image

alignment algorithm serves as the core block for direct visual odometry approaches. It jointly optimizes camera poses, scene structure as well as the camera intrinsic parameters by maximizing the photometric consistency across multiple images. For simplicity, we only consider optimizing over the relative camera pose here, but the approach extends naturally to the full problem. It can be formally defined as follows:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{i=0}^{m-1} \|\mathbf{I}_{\text{ref}}(\mathbf{x}_i) - \mathbf{I}_{\text{cur}}(\hat{\mathbf{x}}_i)\|_2^2, \tag{7.12}$$

where $\mathbf{T} \in \mathbf{SE}(3)$ is the transformation matrix from the reference image $\mathbf{I}_{\text{ref}}$ to the current image $\mathbf{I}_{\text{cur}}$, m is the number of sampled pixels for motion estimation, $\mathbf{x}_i \in \mathbb{R}^2$ is the location of the i^{th} pixel, $\hat{\mathbf{x}}_i \in \mathbb{R}^2$ is the pixel location corresponding to pixel $\mathbf{x}_i$ in current image $\mathbf{I}_{\text{cur}}$. Robust loss function (e.g. huber loss) is usually also applied to the error residuals for robust pose estimation. The image points $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ are related by the camera pose $\mathbf{T}$ and the depth d_i as

$$\hat{\mathbf{x}}_i = \pi(\mathbf{T} \cdot \pi^{-1}(\mathbf{x}_i, d_i)), \tag{7.13}$$

where $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the camera projection function, which projects point in 3D space to image plane; $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the inverse projection function, which transforms a 2D point from image to 3D space by backprojecting with the depth d_i.

Direct VO methods assume that photoconsistency (i.e. equation 7.12) holds for the correct transformation $\mathbf{T}$. However, if the images $\mathbf{I}_{\text{ref}}$ and $\mathbf{I}_{\text{cur}}$ are affected by different motion blur, the photoconsistency loss will no longer be valid since the local appearance for correctly corresponding points will differ. This scenario is unavoidable in settings with highly non-linear trajectories, e.g. tracking in augmented/mixed/virtual reality applications, which usually result in images with different levels of motion blur.

7.3.3. Motion trajectory modeling

To correctly compensate for the motion blur we need to model the local camera trajectory during the exposure time. One approach is to only parameterize the final camera pose and then linearly interpolate between the previous frame and the new estimate. From the interpolation we can then create the virtual images necessary to represent the motion blur, as in equation (7.11). However, this approach might fail for camera trajectories with very abrupt directional changes, which are quite common for hand-held and head-mounted cameras.

To ensure robustness, instead, we choose to parameterize the local camera trajectory independently of the previous frame. To be specific, we parameterize two camera poses, one at the beginning of the exposure $\mathbf{T}_0^w \in \mathbf{SE}(3)$ and one at the end $\mathbf{T}_\tau^w \in \mathbf{SE}(3)$. Between the two poses we linearly interpolate poses in the Lie-algebra of $\mathbf{SE}(3)$. The virtual camera pose at time $t \in [0, \tau]$ can thus be represented as

$$\mathbf{T}_t^w = \mathbf{T}_0^w \cdot \exp(\frac{t}{\tau} \cdot \log((\mathbf{T}_0^w)^{-1} \cdot \mathbf{T}_\tau^w)), \tag{7.14}$$

where τ is the exposure time. For efficiency, we decompose Eq. (7.14) as

$$\bar{\mathbf{q}}_t^w = \bar{\mathbf{q}}_0^w \otimes \exp(\frac{t}{\tau} \cdot \log((\bar{\mathbf{q}}_0^w)^{-1} \otimes \bar{\mathbf{q}}_\tau^w)), \tag{7.15}$$

$$\mathbf{t}_t^w = \mathbf{t}_0^w + \frac{t}{\tau}(\mathbf{t}_\tau^w - \mathbf{t}_0^w), \tag{7.16}$$

where $\mathbf{T}_*^w = [\mathbf{R}_*^w | \mathbf{t}_*^w] \in \mathrm{SE}(3)$, $\mathbf{R}_*^w \in \mathrm{SO}(3)$ and $\mathbf{t}_*^w \in \mathbb{R}^3$. We represent the rotation matrix $\mathbf{R}_*^w$ with unit quaternion $\bar{\mathbf{q}}_*^w$. The goal of our motion blur-aware tracker is now to estimate both $\mathbf{T}_0^w$ and $\mathbf{T}_\tau^w$ for each frame. If the two poses are close, we know that the corresponding frame has very little motion blur. In this work we only considered linear interpolation between the two poses, but e.g. higher order splines could be used as well which could then represent more complex camera motions. However, in our experiments we found that the linear model worked well enough, since the exposure time is usually relatively short.

Local parameterization of rotation: For the real implementation, we use the local parameterization for the update of the rotation. The plus operation for unit quaternion $\bar{\mathbf{q}}$ is defined as

$$\bar{\mathbf{q}}' = \bar{\mathbf{q}} \otimes \Delta\bar{\mathbf{q}}, \tag{7.17}$$

where $\Delta\bar{\mathbf{q}} = \exp(\Delta\mathbf{r})$, $\Delta\mathbf{r} = \begin{bmatrix}\Delta r_x & \Delta r_y & \Delta r_z\end{bmatrix}^T$ and $\Delta r_x \to 0$, $\Delta r_y \to 0$, $\Delta r_z \to 0$. The Jacobian with respect to $\Delta\mathbf{r}$ can thus be derived as

$$\frac{\partial\bar{\mathbf{q}}'}{\partial\Delta\mathbf{r}} = \mathbf{Q}(\bar{\mathbf{q}}) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}. \tag{7.18}$$

Jacobian related to translation: We can simplify Eq. (7.16) as

$$\mathbf{t}_t^w = \frac{\tau - t}{\tau}\mathbf{t}_0^w + \frac{t}{\tau}\mathbf{t}_\tau^w. \tag{7.19}$$

The Jacobians, i.e. $\frac{\partial\mathbf{t}_t^w}{\partial\mathbf{t}_0^w} \in \mathbb{R}^{3\times3}$ and $\frac{\partial\mathbf{t}_t^w}{\partial\mathbf{t}_\tau^w} \in \mathbb{R}^{3\times3}$ can thus be derived as

$$\frac{\partial\mathbf{t}_t^w}{\partial\mathbf{t}_0^w} = \begin{bmatrix} \frac{\tau-t}{\tau} & 0 & 0 \\ 0 & \frac{\tau-t}{\tau} & 0 \\ 0 & 0 & \frac{\tau-t}{\tau} \end{bmatrix}, \tag{7.20}$$

$$\frac{\partial\mathbf{t}_t^w}{\partial\mathbf{t}_\tau^w} = \begin{bmatrix} \frac{t}{\tau} & 0 & 0 \\ 0 & \frac{t}{\tau} & 0 \\ 0 & 0 & \frac{t}{\tau} \end{bmatrix}. \tag{7.21}$$

Jacobian related to rotation: We decompose Eq. (7.15) as

$$\bar{\mathbf{q}}_\tau^0 = (\bar{\mathbf{q}}_0^w)^{-1} \otimes \bar{\mathbf{q}}_\tau^w, \tag{7.22}$$

$$\mathbf{r} = \frac{t}{\tau} \cdot \log(\bar{\mathbf{q}}_\tau^0), \tag{7.23}$$

$$\bar{\mathbf{q}}_t^0 = \exp(\mathbf{r}), \tag{7.24}$$

$$\bar{\mathbf{q}}_t^w = \bar{\mathbf{q}}_0^w \otimes \bar{\mathbf{q}}_t^0. \tag{7.25}$$

We can rewrite both Eq. (7.22) and Eq. (7.25) as

$$\bar{\mathbf{q}}_\tau^0 = \mathbf{Q}((\bar{\mathbf{q}}_0^w)^{-1}) \cdot \bar{\mathbf{q}}_\tau^w = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_\tau^w) \cdot (\bar{\mathbf{q}}_0^w)^{-1}, \tag{7.26}$$

$$\bar{\mathbf{q}}_t^w = \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \bar{\mathbf{q}}_t^0 = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_t^0) \cdot \bar{\mathbf{q}}_0^w. \tag{7.27}$$

The Jacobian $\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} \in \mathbb{R}^{4\times4}$ can thus be derived as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_t^0) + \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \bar{\mathbf{q}}_0^w}, \tag{7.28}$$

$$\frac{\partial \bar{\mathbf{q}}_t^0}{\partial \bar{\mathbf{q}}_0^w} = \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0} \cdot \hat{\mathbf{Q}}(\bar{\mathbf{q}}_\tau^w) \cdot \frac{\partial (\bar{\mathbf{q}}_0^w)^{-1}}{\partial \bar{\mathbf{q}}_0^w}. \tag{7.29}$$

Similarly for the Jacobian $\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} \in \mathbb{R}^{4\times4}$, we can derive it as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} = \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0} \cdot \mathbf{Q}((\bar{\mathbf{q}}_0^w)^{-1}). \tag{7.30}$$

Note that both $\frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}}$ and $\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0}$ are the Jacobians related to the exponential mapping and logarithm mapping respectively. $\frac{\partial (\bar{\mathbf{q}}_0^w)^{-1}}{\partial \bar{\mathbf{q}}_0^w}$ is the Jacobian related to the inverse of quaternion.

The Jacobians with respect to the local parameterization can then be computed as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \Delta\mathbf{r}_0^w} = \frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} \cdot \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}, \tag{7.31}$$

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \Delta\mathbf{r}_\tau^w} = \frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} \cdot \mathbf{Q}(\bar{\mathbf{q}}_\tau^w) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}. \tag{7.32}$$

7.3.4. Direct image alignment with blurry images

Our motion blur-aware tracker works by performing direct alignment between the keyframe, which we assume is sharp, and the current frame which can suffer from motion blur. To leverage photometric consistency in the alignment, we thus need to either de-blur the new frame or re-blur the keyframe. In our work we chose the latter since re-blurring is in general easier and more robust compared to motion deblurring, especially for severe motion blurred images.

Each sampled pixel in $\mathbf{I}_{\text{ref}}$ with known depth can be transferred into the current (blurry) image $\mathbf{B}_{\text{cur}}$ using (7.13). For each projected point we select its nearest neighbour integer position pixel, in current blurry image. Assuming that the 3D point lies on a fronto-parallel plane (with respect to $\mathbf{I}_{\text{ref}}$), we can use this plane to transfer the selected pixel back into the reference view. Details can be found in Fig. 7.2. To synthesize the re-blurred pixel from the reference view (so that we can compare against the real captured pixel intensity), we now interpolate between $\mathbf{T}_0^w$ and $\mathbf{T}_\tau^w$. For each virtual view $\mathbf{T}_t^w$, which is uniformly sampled within $[0, \tau]$, we transfer the pixel coordinate (i.e. the red pixel in Fig. 7.2) back into the reference image and retrieve the image intensity values using bi-linear interpolation. The re-blurred pixel intensity is then created by averaging over the intensity values (as in (7.11)):

$$\hat{\mathbf{B}}_{\text{cur}}(\mathbf{x}) = \frac{1}{n}\sum_{i=0}^{n-1} \mathbf{I}_{\text{ref}}(\mathbf{x}_{\frac{i\tau}{n-1}}), \tag{7.33}$$

where $\mathbf{x}_{\frac{i\tau}{n-1}} \in \mathbb{R}^2$ corresponds to the transferred point at time $t = \frac{i\tau}{n-1}$ in the sharp reference frame. The tracker then optimizes over the start-pose and end-pose to minimize the photoconsistency loss between the real captured intensities in current frame and the synthesized pixel intensities from the reference image (i.e. via re-blurring),

$$\mathbf{T}_0^{w*},\ \mathbf{T}_\tau^{w*} = \underset{\mathbf{T}_0^w,\,\mathbf{T}_\tau^w}{\operatorname{argmin}} \sum_{i=0}^{m-1} \left\| \mathbf{B}_{\text{cur}}(\mathbf{x}_i) - \hat{\mathbf{B}}_{\text{cur}}(\mathbf{x}_i) \right\|_2^2. \tag{7.34}$$

In practice, most direct image alignment methods use local patches for

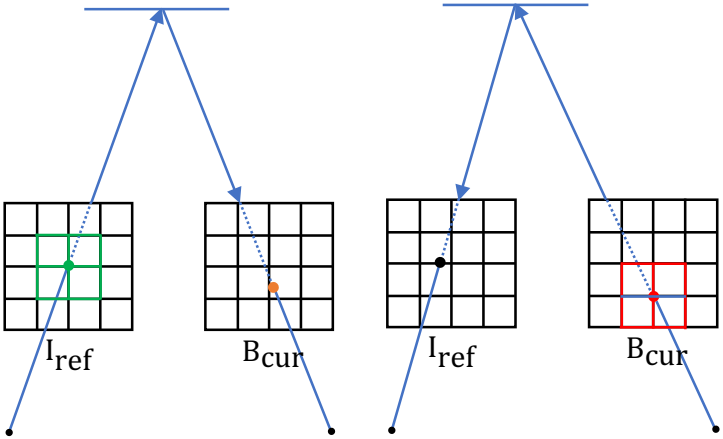


Figure 7.2.: Pixel point transfer strategies. Note that we assume the pixel center lies at the grid intersection, e.g. the green grid is considered as a 3×3 patch.

better convergence. Different from direct image alignment algorithm for sharp images, which usually selects the local patch from the reference image (e.g. the green 3×3 grid on the left of Fig. 7.2), we instead select the local patch from the current blurry image (e.g. the red 3×3 grid on the right of Fig. 7.2) since this simplifies the re-blurring step of our pipeline.

7.3.5. More details on the transfer

To further demonstrate the relationship between $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{x}_{\frac{i\tau}{n-1}} \in \mathbb{R}^2$ from Eq. (7.33), we define following notations for the ease of illustration. We denote the depth of the fronto-parallel plane as d, which is the estimated depth of the corresponding sampled keypoint from $\mathbf{I}_{\text{ref}}$ (i.e. the green pixel in Fig. 7.2); we further denote the camera pose of the virtual frame $\mathbf{I}_i$ captured at timestamp $\frac{i\tau}{n-1}$ relative to the reference keyframe $\mathbf{I}_{\text{ref}}$ as

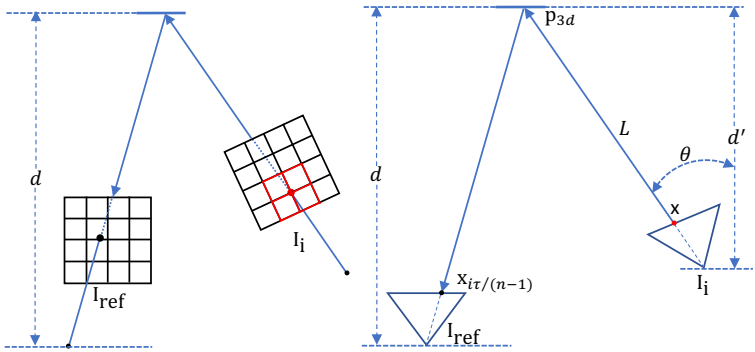


Figure 7.3.: Geometric relationship between $\mathbf{x} \in \mathbb{R}^2$ (i.e. the red pixel) of the virtual sharp image $\mathbf{I}_i$ and $\mathbf{x}_{\frac{i\tau}{n-i}} \in \mathbb{R}^2$ (i.e. the black pixel) of the reference image $\mathbf{I}_{\text{ref}}$.

$\mathbf{T}_i^{ref} \in \mathbf{SE}(3)$, which can be computed from Eq. (7.14) as

$$\mathbf{T}_i^{ref} = \mathbf{T}_0^{ref} \cdot \exp(\frac{i}{n-1}\tau \cdot log(\mathbf{T}_0^{ref\,-1} \cdot \mathbf{T}_\tau^{ref})), \tag{7.35}$$

where $\mathbf{T}_0^{ref} \in \mathbf{SE}(3)$ and $\mathbf{T}_\tau^{ref} \in \mathbf{SE}(3)$ are the camera poses (which are defined from the camera coordinate frame to the reference coordinate frame) of the current blurry image, at the beginning and end of the image capturing respectively, τ is the camera exposure time. Different from Eq. (7.14), which uses global poses (i.e. $\mathbf{T}_0^w$ and $\mathbf{T}_\tau^w$) to parameterize the local camera trajectory, we use the relative poses (i.e. $\mathbf{T}_0^{ref}$ and $\mathbf{T}_\tau^{ref}$) instead for the real implementations of our tracker. The global camera pose can then be obtained from the known reference camera pose $\mathbf{T}_{ref}^w$. Note that the fronto-parallel plane is defined in the reference camera frame, it might not be fronto-parallel with respect to the i^{th} virtual camera frame. To avoid confusion, we illustrate the relationship in Fig. 7.3.

We denote the translation vector $\mathbf{p}_i^{ref}$ of $\mathbf{T}_i^{ref}$ with $[p_x, p_y, p_z]^T$ and represent the rotation $\mathbf{R}_i^{ref}$ with unit quaternion, i.e. $\bar{\mathbf{q}} = [q_x, q_y, q_z, q_w]^T$.

We denote d as the depth of the frontal-parallel plane with respect to the reference key-frame. We can then compute the distance d' between the camera center of the i^{th} virtual camera to the frontal-parallel plane as

$$d' = d - p_z. \tag{7.36}$$

The unitary ray of pixel $\mathbf{x} \in \mathbb{R}^2$ in the i^{th} image $\mathbf{I}_i$ can be computed by the back-projection function $\pi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \pi^{-1}(\mathbf{x}), \tag{7.37}$$

where $x^2 + y^2 + z^2 = 1$. We can then compute cosine function of the angle (i.e. θ) between the unitary ray and the plane normal of the frontal parallel plane as

$$\lambda = \cos(\theta) = (\mathbf{R}_i^{ref} \cdot \pi^{-1}(\mathbf{x}))^T \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{7.38}$$

from which we can further simply it as

$$\lambda = 2x(q_x q_z - q_w q_y) + 2y(q_x q_w + q_y q_z) + z(q_w^2 - q_x^2 - q_y^2 + q_z^2). \tag{7.39}$$

The length of the line segment L, which goes through pixel point $\mathbf{x}$ from camera center of the i^{th} camera and intersects with the frontal-parallel plane, can then be simply computed as

$$|L| = \frac{d'}{\lambda} = \frac{d - p_z}{\lambda}. \tag{7.40}$$

The 3D intersection point $\mathbf{p}_{3d}$ between the line segment L and the frontal parallel plane can thus be computed as

$$\mathbf{p}_{3d} = |L| \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{d - p_z}{\lambda} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{7.41}$$

where the $\mathbf{p}_{3d}$ is represented in the coordinate frame of the i^{th} camera. To compute the corresponding pixel point $\mathbf{x}_{\frac{i\tau}{n-1}}$ in the reference image $\mathbf{I}_{\text{ref}}$, we need transform the 3D point $\mathbf{p}_{3d}$ to the reference camera coordinate frame and then project it to the image plane. It can be formally defined as

$$\mathbf{p}'_{3d} = \mathbf{T}_i^{ref} \cdot \mathbf{p}_{3d}, \tag{7.42}$$

$$\mathbf{x}_{\frac{i\tau}{n-1}} = \pi(\mathbf{p}'_{3d}), \tag{7.43}$$

where $\mathbf{p}'_{3d}$ is the 3D point $\mathbf{p}_{3d}$ represented in the reference camera coordinate frame, $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the camera projection function.

Jacobian derivations: The pose of the i^{th} virtual camera, i.e. $\mathbf{T}_i^{ref}$, relates to $\mathbf{T}_0^{ref}$ and $\mathbf{T}_\tau^{ref}$ via Eq. (7.35). To estimate both $\mathbf{T}_0^{ref}$ and $\mathbf{T}_\tau^{ref}$, we need to have the Jacobian of $\mathbf{x}_{\frac{i\tau}{n-1}}$ with respect to $\mathbf{T}_i^{ref}$. Since the relationship between $\mathbf{x}_{\frac{i\tau}{n-1}}$ and $\mathbf{T}_i^{ref}$ is complex, as derived above, we use the Mathematica Symbolic Toolbox[1] for the ease of Jacobian derivations. The details are as follows.

$$\alpha_0 = q_x x + q_y y + q_z z, \qquad \alpha_1 = q_y x - q_w z - q_x y, \tag{7.44}$$

$$\alpha_2 = q_w y - q_x z + q_z x, \qquad \alpha_3 = q_w x + q_y z - q_z y, \tag{7.45}$$

$$\alpha_4 = q_w z + q_x y - q_y x, \qquad \beta_0 = -2(q_w q_z - q_x q_y), \tag{7.46}$$

$$\beta_1 = 2(q_w q_y + q_x q_z), \qquad \beta_2 = 2(q_w q_z + q_x q_y), \tag{7.47}$$

$$\beta_3 = -2(q_w q_x - q_y q_z), \qquad \beta_4 = -2(q_w q_y - q_x q_z), \tag{7.48}$$

$$\beta_5 = 2(q_w q_x + q_y q_z), \tag{7.49}$$

$$\gamma_0 = x(q_w^2 + q_x^2 - q_y^2 - q_z^2) + y\beta_0 + z\beta_1, \tag{7.50}$$

$$\gamma_1 = x\beta_2 + y(q_w^2 - q_x^2 + q_y^2 - q_z^2) + z\beta_3, \tag{7.51}$$

$$\gamma_2 = x\beta_4 + y\beta_5 + z(q_w^2 - q_x^2 - q_y^2 + q_z^2), \tag{7.52}$$

[1] https://www.wolfram.com/mathematica/

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{7.53}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{7.54}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_z} = \begin{bmatrix} -\gamma_0/\lambda \\ -\gamma_1/\lambda \\ 1-\gamma_2/\lambda \end{bmatrix}, \tag{7.55}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_x} = 2\frac{d-p_z}{\lambda} \begin{bmatrix} \alpha_0 - \alpha_2\gamma_0/\lambda \\ \alpha_1 - \alpha_2\gamma_1/\lambda \\ \alpha_2 - \alpha_2\gamma_2/\lambda \end{bmatrix}, \tag{7.56}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_y} = 2\frac{d-p_z}{\lambda} \begin{bmatrix} \alpha_4 + \alpha_3\gamma_0/\lambda \\ \alpha_0 + \alpha_3\gamma_1/\lambda \\ -\alpha_3 + \alpha_3\gamma_2/\lambda \end{bmatrix}, \tag{7.57}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_z} = 2\frac{d-p_z}{\lambda} \begin{bmatrix} -\alpha_2 + \alpha_0\gamma_0/\lambda \\ \alpha_3 - \alpha_0\gamma_1/\lambda \\ \alpha_0 - \alpha_0\gamma_2/\lambda \end{bmatrix}, \tag{7.58}$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_w} = 2\frac{d-p_z}{\lambda} \begin{bmatrix} \alpha_3 - \alpha_4\gamma_0/\lambda \\ \alpha_2 - \alpha_4\gamma_1/\lambda \\ \alpha_4 - \alpha_4\gamma_2/\lambda \end{bmatrix}. \tag{7.59}$$

The Jacobian $\frac{\partial \mathbf{x}_{\frac{i\tau}{n-1}}}{\partial \mathbf{p}'_{3d}} \in \mathbb{R}^{2\times 3}$ is related to the camera projection function. For a pinhole camera model with the intrinsic parameters f_x, f_y, c_x, c_y, it can be derived as

$$\frac{\partial \mathbf{x}_{\frac{i\tau}{n-1}}}{\partial \mathbf{p}'_{3d}} = \begin{bmatrix} \frac{f_x}{\mathbf{p}'_{3d_z}} & 0 & -\frac{f_x \cdot \mathbf{p}'_{3d_x}}{(\mathbf{p}'_{3d_z})^2} \\ 0 & \frac{f_y}{\mathbf{p}'_{3d_z}} & -\frac{f_y \cdot \mathbf{p}'_{3d_y}}{(\mathbf{p}'_{3d_z})^2} \end{bmatrix}, \tag{7.60}$$

where $\mathbf{p}'_{3d} = [\mathbf{p}'_{3d_x}, \mathbf{p}'_{3d_y}, \mathbf{p}'_{3d_z}]^T$.

7.4. Datasets

In this section we give an overview of the datasets that we consider in our experimental evaluation. While there are many different datasets for evaluating visual odometry methods, we found that there is no suitable dataset that specifically targets at motion blurred images, although some datasets have sub-sequences that contain motion blur, e.g. in the ETH3D SLAM Benchmark [SSP19] and TUM RGB-D [SEE+12].

ETH3D [SSP19] / ArchVizInterior: In the ETH3D SLAM benchmark [SSP19], the image sequences; *camera_shake_1*, *camera_shake_2* and *camera_shake_3* have severe motion blur. The three sequences were captured with a camera being quickly shaken back and forth. In addition to the motion blur, the sequences are difficult due to the very poorly textured scene (mainly containing a white circular table with very few distinguishing landmarks). We experiment with both DSO [EKC17] and ORBSLAM [MAT17a] on these sequences and find that both methods fail to initialize on this dataset.

To investigate if the failures are due to the poorly textured scene or the motion blur, we render a synthetic photo-realistic dataset using the same motion trajectories, exposure time and frame rate. The dataset is rendered by the Unreal game engine with the free ArchVizInterior scene model[2]. We use the scripts provided by Liu et al. [LCLP20] for the dataset creation.

[2] https://www.unrealengine.com

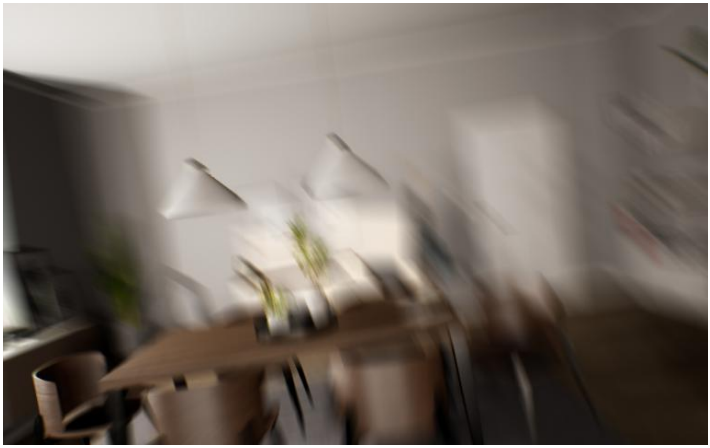


Figure 7.4.: *Sample image from our synthetic ArchVizInterior dataset*. The camera motions are taken from the ETH3D Benchmark [SSP19] which are then re-rendered in synthetic scene by Unreal game engine.

A sample image can be found in Fig. 7.4. Sample videos on the dataset can be found in our supplementary material. Since this dataset provides perfect ground truth sharp images, which are paired with the motion blurred images, we use it for our ablation studies.

TUM-RGBD [SEE+12]: The hand-held SLAM sequences from the TUM-RGBD dataset [SEE+12] also contain motion blurred images. The dataset is collected with the Microsoft Xbox Kinect sensor, which contains a rolling shutter color camera and a time-of-flight depth camera. Since the dataset targets at evaluating the performance of RGBD camera based SLAM methods, the effect of motion blurred images is not their focus. Furthermore, it has been shown that direct approach is more sensitive to rolling shutter effect [SSP19, YWGC18]. It is thus better to have dataset which is collected from a global shutter camera to avoid the effect of rolling shutter mechanism. We also evaluate our method with the TUM-RGBD dataset in the next section.

Figure 7.5.: Examples images from the proposed dataset for benchmarking visual odometry from motion blurred image sequences. The dataset contains multiple sequences with varying levels of motion blur.

Proposed Motion Blur Benchmarking Dataset: To more clearly show the benefit of our method we propose a new benchmark dataset for evaluating visual odometry which specifically targets motion blur. By making this dataset publicly available to other researchers, we hope to encourage further research on making visual odometry robust.

Our dataset is collected with a global shutter camera at a resolution of 752×480 pixels and a frame rate of 27 fps. The ground truth trajectory is provided by an indoor motion capturing system[3] at 100 Hz. Extrinsic parameters between the marker for motion capture and camera is calibrated by the hand-eye calibration approach, such that the ground truth trajectory can align with the camera motion trajectory. A total number of 18 motion blurred sequences are collected. The dataset contains images with varying levels of motion blur. More details on the dataset can be found in the supplementary video. Figure 7.5 shows some example images from the new dataset with varying motion blur.

7.5. Experimental evaluation

Implementation details: The original tracker of DSO [EKC17] does semi-dense direct image alignment. For efficiency, we sub-sample the high gradient pixels to obtain sparse keypoints, which are uniformly distributed

[3] https://www.vicon.com

within the image. We further define a 9×9 local patch around each sampled sparse keypoints for better convergence. The energy function is optimized in a coarse to fine manner and robust huber loss function is also applied for robustness. Our tracker is implemented and evaluated on a laptop grade Nvidia RTX 2080 graphic card. It takes 34.4 ms on average to process a single blurry image, which is suitable for real time applications.

We experiment with two state-of-the-art deblurring networks, SRNDeblurNet [TGS+18] and DeblurGANv2 [KMWW19]. We use the official pretrained models and generalize them to our datasets without any finetuning. In particular, we consider the mobile network of DeblurGANv2 since it can run in real time on a high-end GPU. SRNDeblurNet takes around 140 ms second to process a single image at a resolution of 752×480 pixels on a laptop grade Nvidia RTX 2080 graphic card. However, it delivers higher quality deblurred images compared to the DeblurGANv2 mobile network and the time consumption is already sufficient for local mapping. There are also more advanced deblurring networks being proposed recently, such as the work from Gao et al. [GTSJ19]. However, they are usually more time consuming than SRNDeblurNet [TGS+18], and is not suitable to be integrated into our local mapper.

Baseline methods and evaluation metrics: Two state-of-the-art monocular VO pipelines are selected for the benchmark. In particular, we select ORB-SLAM [MAT17a], which is the representative of sparse feature based approach. As a representative for direct approaches we compare with DSO [EKC17]. For quantative comparisons, we measure the RMSE of absolute trajectory error (i.e. RMSE ATE) [SEE+12], since it is the focus of VO algorithms and is commonly used by the literature [EKC17, SSP19]. The estimated trajectory is first aligned with the ground truth by matching poses with the same timestamps. The RMSE of ATE is then computed by averaging the translational differences between the aligned trajectories. In addition, we also use the percentage of frame drops to measure the robustness of different algorithms.

Motivating example: To clearly motivate the need for motion blur aware VO, we first evaluate the performance of ORB-SLAM and DSO on the sharp images and the corresponding motion blurred images respectively, on the

ArchVizInterior dataset. See Section 7.4 for details on the dataset. Table 7.1 demonstrates that motion blurred images affect both ORB-SLAM and DSO, in terms of estimated trajectory accuracy and robustness. Although there is no large accuracy drop for ORB-SLAM, there are significant frame drops. ArchVizInterior dataset collects images around a local area and scene overlaps exist among almost every image. Even though there are many frame drops, ORB-SLAM can still recover back due to its relocalization module, once the image is in better quality, assuming there is enough visual overlap with previously mapped areas. However, a reset might need to perform if the camera tranverses in un-explored scenes. There is no frame drops for DSO. However, its accuracy drops with a large margin compared to that with sharp images.

Note that we rendered the dataset with the same trajectories from ETH3D [SSP19] dataset. The results demonstrate that the camera motion is not the main factor, which leads the failure of both ORB-SLAM and DSO on ETH3D dataset. It further justifies our motivation to create new datasets.

Ablation studies: Since ArchvizInterior dataset has ground truth sharp images, which are paired with the motion blurred images, we conduct ablation studies with it for better comparisons. Our ablation studies consist of two parts, the selection of the deblurring network and experiments to demonstrate the effectiveness of our motion blur aware tracker.

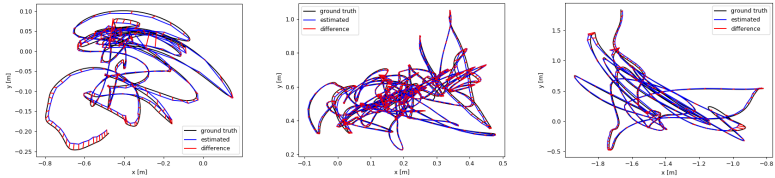
We evaluate the generalization performance as well as efficiency of both deblurring networks, i.e. SRNDeblurNet [TGS+18] and DeblurGANv2-mobileNet [KMWW19] on the ArchvizInterior dataset. The evaluation is conducted with a laptop grade Nvidia RTX 2080 graphic card. Table 7.2 demonstrates that the DeblurGANv2-mobileNet is able to run in real time on a high-end GPU. However, its deblurring performance is worse than SRNDeblurNet as an expense. To verify if current performance of DeblurGANv2-mobileNet is sufficient to improve the performance of VO algorithms, we deblurred every image of ArchVizInterior dataset by DeblurGANv2-mobileNet. We run both ORB-SLAM and DSO with the deblurred images. The experimental results shown in Table 7.1 demonstrate that it can only improve the performance of VO algorithms with motion blurred images with a small margin. The reason is that the Deblur-

	ORB-SLAM [MAT17a]						DSO [EKC17]					
	ArchViz-1		ArchViz-2		ArchViz-3		ArchViz-1		ArchViz-2		ArchViz-3	
	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)
Sharp	0.0201	0	0.0048	0	0.0138	0	0.0196	0	0.0043	0	0.0140	0
Blur	0.0325	22.1	0.0122	1.068	0.1008	19.5	0.2132	0	0.1655	0	0.1286	0
Deblur	0.0179	15.6	0.0066	2.763	0.0197	16.7	0.2065	0	0.1613	0	0.0481	0

Table 7.1.: *The performance of both ORB-SLAM and DSO on the ArchVizInterior dataset*. *Sharp*, *Blur* and *Deblur* denote the pipeline is running on the ground truth sharp images, motion blurred images and the deblurred images by DeblurGANv2 [KMWW19] respectively. The FD column shows the percentage of dropped frames. Both the ATE and FD metrics are the smaller the better.

	PSNR (dB) ↑	SSIM ↑	Time (ms)
Blur image	26.80	0.7887	N.A.
DeblurGANv2m [KMWW19]	28.66	0.8156	38.1
SRNDeblurNet [TGS+18]	30.01	0.8491	140.3

Table 7.2.: Generalization performance of DeblurGANv2-mobileNet [KMWW19] and SRNDeblurNet [TGS+18] on the ArchVizInterior dataset.



ArchViz-1 ArchViz-2 ArchViz-3

Figure 7.6.: Estimated trajectories of MBA-VO from the motion blurred image sequences of the ArchVizInterior dataset. It demonstrates that MBA-VO can estimate accurate trajectories, although the camera motions are very challenging.

GANv2 [KMWW19] has limited generalization performance as an expense for smaller model size. It demonstrates that the naive way to deblur every input frame (with an efficient deblurring network for real time operation), and feed the deblurred images to a standard VO pipeline is not the correct way to make VO robust to motion blur. It justifies our motivation to do hybrid motion blur aware VO, which can take advantage of a more powerful deblurring network with a larger model size. Our hybrid approach recovers the camera motion of severe blurred images by the motion blur aware direct image alignment algorithm, without the need to deblur them in frame rate. Since SRNDeblurNet takes around 140 ms to process a 752×480 pixels resolution image, which is sufficient to deblur selected key-frame image, and delivers better deblurring performance, we thus use it for our pipeline.

	ArchViz-1		ArchViz-2		ArchViz-3	
	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)
Sharp	0.0197	0	0.0101	0	0.0157	0
Blur	0.0256	0	0.0184	0	0.0202	0

Table 7.3.: *The performance of MBA-VO on the ArchVizInterior dataset*. *Sharp* and *Blur* denote the pipeline is running on the ground truth sharp images and motion blurred images respectively. FD is the percentage of dropped frames.

To study the effectiveness of MBA-VO, we experiment with sharp images and blurry images respectively. Experimental results from Table 7.3 demonstrate that MBA-VO is able to achieve similar performance as both ORB-SLAM and DSO if the images are not motion blurred. For motion blurred images, MBA-VO is able to achieve competitive accuracy as that for sharp images without any frame drops. To further demonstrate the effectiveness of our motion blur aware tracker, we set the camera exposure time to be 0 during the estimation of the camera poses (i.e. Eq. (7.34)). It enforces the tracker to assume the current blurry images as sharp images and do normal pose estimations instead. The other settings are kept the same (e.g. we still use SRNDeblurNet to deblur the keyframe images). The resulted ATE metrics are 0.22 m, 0.1558 m and 0.2113 m respectively for the ArchVizInterior dataset. The experimental results thus demonstrate the necessity to do motion blur aware tracking.

Fig. 7.6 demonstrates the estimated trajectories of MBA-VO on the motion blurred sequences from ArchVizInterior dataset. Both the quantitative and qualitative results demonstrate the effectiveness of our proposed algorithm for motion blurred image sequences.

Evaluation with TUM RGB-D dataset: To evaluate the performance of MBA-VO with real motion blurred images, we select three sequences with large motion blur from the TUM RGB-D dataset [SEE+12]. In particular, we select the *fr1-desk*, *fr1-desk2* and *fr1-room* from the handheld

	fr1-desk		fr1-desk2		fr1-room	
	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)
ORBSLAM	0.1781	5.1	0.3005	33.8	0.0657	46.5
DSO	0.4956	0	0.7762	0	0.2992	0
MBA-VO	0.1021	0	0.3997	0	0.1435	0

Table 7.4.: *Comparison on TUM RGB-D dataset [SEE+12].* ORB-SLAM suffers from significant frame drops, although it provides accurate estimates. The proposed method, MBA-VO, improves on DSO and provides more accurate estimates with no frame drops.

SLAM category. Since the camera is hand held, which is similar to head-mounted camera, hand shaken would result in fast rotational motion though the translational velocity is small. For augmented/virtual/mixed reality applications, head rotational motion is the main cause of severe motion blur.

Table 7.4 demonstrates the performance of MBA-VO against ORB-SLAM and DSO on sequences with large motion blur from TUM RGB-D dataset. It demonstrates that MBA-VO is able to improve the accuracy over the original DSO algorithm, while is also more robust compared to sparse feature based approach, with motion blurred images. Note that ORB-SLAM suffers from significant frame-drops, although it generally provides more accurate poses for these sequences (low ATE).

Evaluation with our real-world dataset: Since the goal of the TUM RGB-D dataset is not evaluating the robustness of monocular VO/SLAM algorithms, we created a specific large real dataset with varying levels of motion blur (see Section 7.4). We evaluate ORB-SLAM, DSO and MBA-VO with it. Table 7.5 presents experimental results with our real-world dataset. It demonstrates that ORB-SLAM [MAT17a] suffers from significant frame drops if the images are motion blurred. It is reasonable since ORB-SLAM is a sparse feature based approach. If the images are severely motion blurred, the sparse feature detector would have difficulties to detect

	ORB-SLAM [MAT17a]		DSO [EKC17]		MBA-VO	
	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)
Seq0	0.1265	7.0	0.2722	0	0.0581	0
Seq1	0.0839	36.8	0.4327	0	0.0692	0
Seq2	0.1992	11.9	0.1958	0	0.0446	0
Seq3	x	x	0.4044	0	0.1615	0
Seq4	x	x	x	x	0.1323	0
Seq5	0.1931	73.2	0.3241	0	0.2667	0
Seq6	0.0743	25.4	0.4968	0	0.3321	0
Seq7	0.1872	47.9	0.3857	0	0.2718	0
Seq8	0.5861	31.9	0.7906	0	0.3915	0
Seq9	0.3791	26.6	0.8538	0	0.2838	0
Seq10	0.1708	33.6	0.4800	0	0.4319	0
Seq11	0.1378	39.1	x	x	0.4003	0
Seq12	x	x	0.5031	0	0.3632	0
Seq13	x	x	0.4501	0	0.3043	0
Seq14	x	x	x	x	0.4516	0
Seq15	x	x	x	x	0.3687	0
Seq16	x	x	x	x	0.3765	0
Seq17	x	x	x	x	0.3299	0

Table 7.5.: *The performance of MBA-VO on our dataset*. x denotes the corresponding algorithm fails on that particular sequence.

enough good features for motion estimation. In contrast, DSO [EKC17] is more robust to motion blur (i.e. no frame drops). However, since pairs of motion blurred images usually violate the photometric-consistency assumption, the accuracy of DSO is degraded. Our proposed motion blur aware visual odometry (i.e. MBA-VO) models the motion blur for direct image alignment algorithm, so that the photometric-consistency assumption would still hold even the images are motion blurred. It thus achieves better accuracy and robustness compared to DSO and ORB-SLAM.

Performance in the absence of motion-blur: To further demonstrate the performance of our proposed method with images in good quality, we run an experiment comparing the proposed approach with DSO and

	ORB-SLAM	DSO	MBA-VO
MH_01_easy	**0.030**	0.050	0.035
MH_02_easy	**0.022**	0.077	0.101
MH_03_medium	**0.049**	0.178	0.239
MH_04_difficult	2.472	1.181	**0.476**
MH_05_difficult	4.386	1.261	**0.265**

Table 7.6.: **EuRoC dataset:** Comparison in terms of ATE RMSE error metric (m). Note that we did not discard any images (e.g. the first few shaky images which are used to initialize IMU) from the EuRoC dataset for our evaluations. Therefore, the resulted accuracy for ORB-SLAM and DSO might be a bit different from prior reported results.

ORB-SLAM on the EuRoC dataset [BNG+16]. Table 7.6 demonstrates that MBA-VO performs slightly worse than DSO on MH_02_easy and MH_03_medium. However, it performs better than DSO on MH_01_easy, MH_04_difficult and MH_05_difficult. In general, we observed similar performance as DSO for good images. The VICON room sequences in the EuRoC dataset are very challenging. Both DSO and MBA-VO fail even with sharp images (either with very large errors or complete tracking failure). This might be caused by the lack of good texture for some of the frames. The sequences also contain degenerate motions (e.g. close to pure rotation). Without any relocalization module (e.g. as is used in ORB-SLAM), it is hard for both DSO and MBA-VO to recover once the pipeline breaks or drifts significantly.

Computational complexity: To compensate the effect of motion blur, we need to sample multiple discrete virtual frames to synthesize the blurry image (Eq.6 in the main text). The number of discrete frames should be larger than the blur kernel size. Currently, we use $n = 64$ for our experiments. If $n = 1$, it reduces to the problem of assuming the images are sharp. The accuracy would thus be affected for motion blurred images. If we raise n to 2 or higher, the number of pose parameters would not be

affected (i.e. 12 variables). However, the number of pixel transfers would be increased (e.g. doubled for 2 virtual frames). Larger n thus requires more computational resources. The back-end is the same as the original DSO [EKC17] and it is running on CPU. Note that the back-end only relies on the deblurred keyframes when optimizing the keyframe poses and structure, thus we can use the original DSO implementation.

Discussions: The experimental results demonstrate that DSO [EKC17] generally performs worse than ORB-SLAM [MAT17a] for motion blurred images, in terms of the ATE metric. It is caused by the datasets we evaluated on have many more severely blurred images. In this case, ORB-SLAM [MAT17a] simply discards the severe blurred images without affecting the accuracy of the remaining frames. In contrast, DSO [EKC17] does not drop the frames, leading to the overall loss of accuracy due to including more challenging frames in the estimation. Note that the ATE metric is only computed from the successfully tracked frames.

7.6. Conclusion

We present a hybrid visual odometry algorithm which is robust to motion blur. The camera motion trajectory within exposure time is explictly modelled and estimated during tracking. It allows us to compensate the effect of motion blur without deblurring all of them. We also propose a novel benchmarking dataset targeting motion blur aware visual odometry. Experimental results demonstrate that our algorithm improves the accuracy and robustness over existing methods on both synthetic and real-world datasets. While we only consider monocular VO, our approach could also be applied to other settings such as VIO and RGB-D SLAM. We believe both our method and dataset would be a valuable step towards the era of robust visual odometry.

8. Is Single Image Motion Estimation Possible?

8.1. Introduction

As the recent advance of deep learning techniques in many vision tasks, several pioneering learning based pipelines have been proposed to tackle the visual odometry problem, with the aim to explore a new research direction to further improve the performance (e.g. in terms of both accuracy and robustness) over existing VO methods. Existing learning based methods can be categorized into end-to-end direct regression approaches and hybrid approaches. Direct regression approaches usually formulate the problem to recover the dense depth maps and relative camera poses via deep networks [ZBSL17, UZU+17]. The networks are trained either in a supervised manner or via self-supervision. Hybrid approaches still adopt the standard formulation of classic VO pipeline, they instead replace parts of the pipeline with deep networks to improve their performance for challenging scenarios [BCC+18a, TTLN17b, OTFY18, BLC+19, YWSC18b].

Although many state-of-the-art end-to-end direct regression approaches have been proposed over the past years, they are still infancy compared to classic approaches in terms of scalability and generalization. In this chapter, we propose to do an in-depth study of those approaches and provide an insight about their characteristics and limitations, such that future research could benefit from. In particular, we focus our attention on the SfMLearner method [ZBSL17] and its variants [YS18, BLW+19]. SfMLearner is a pioneering network to learn the dense depth maps and relative camera motions from a short sequence of images. It consists of two sub-networks, i.e., a dense depth network and a motion estimation network. The depth

network estimates the dense maps from a single image and the motion estimation network predicts the relative pose from a 5-frame snippet. Both networks are trained in a self-supervised manner via cross-view photometric consistency. Since Dijk et al. [DC19] recently propose a work to study the depth network of SfMLearner, we thus further limit our study mainly on the motion estimation network.

Our study is initially driven by the finding that a single image VO is feasible, and achieves competitive performance on the KITTI dataset [GLU12]. Our network has similar architecture as that of prior works [ZBSL17, BLW+19]. Instead of using multiple frames for the motion estimation network as prior methods, we enforce the motion estimation network to use a single image as input and to predict the relative pose from its next consecutive frame to current frame. The whole network is trained in a self-supervised manner with the aid of the next image. Experimental results on KITTI dataset [GLU12] demonstrate that it can achieve competitive performance on par with the other multi-frame learning based approaches [ZBSL17]. In particular, we evaluate the depth estimation on KITTI raw dataset using Eigen et al.'s split. The experimental results demonstrate that our network performs better than two supervised methods, i.e., Eigen et al. [EPF14] and Liu et al. [LSLR16], and an unsupervised method from Zhou et al. [ZBSL17]. We also evaluate the relative pose estimation on KITTI odometry dataset and find that it performs better than ORB-SLAM (short) [MAT17b] and has competitive performance as other unsupervised methods.

How is this possible? In general, it is not possible to predict motion from only a single image. We hypothesize that it is because the KITTI dataset [GLU12] is an "easy" dataset for those learning based approaches. To verify our hypothesis, we train our single image motion estimation network with ground truth motions from KITTI odometry dataset [GLU12], so that we can have an understanding about its upper bound performance. Experimental results demonstrate that the network can really learn the motion directions based on the semantic information possessed by only a single image from the KITTI dataset. For example, the street directions can already tell many motion cues due to the restricted camera motions from the dataset.

To further verify our hypothesis that KITTI dataset is an "easy" dataset for those networks, we create a large dataset with Carla simulator [DRC+17]. We modify the simulator such that the camera can be placed at arbitrary positions and is able to undergo random 6-DoF motions. We train one of the latest best performing state-of-the-art networks, i.e., the network from Bian et al. [BLW+19], with this dataset and evaluate it against a simple two-view structure from motion baseline, which is implemented in MATLAB by ourselves. Experimental results demonstrate the network cannot perform well on this dataset, which has complex camera motions. It is impossible even for us human beings to tell the motion direction from a single image from this dataset, since the camera can move in arbitrary direction. We argue that although existing methods achieve remarkable performance on KITTI dataset [GLU12], there is still much room for those end-to-end learning based approaches to improve, and it is also necessary to use new novel real datasets with complex motions for better evaluations in future.

8.2. Single image motion estimation network

Network architecture: We propose a network architecture which is able to do single image visual odometry on KITTI dataset [GLU12]. Our network architecture is similar as that of Bian et al. [BLW+19] and the detailed network architecture is shown in Fig. 8.1. The network consists of a single image depth estimation network and single image motion estimation network. We use DispNet [MIH+16] for our depth network. The motion estimation network consists of a ResNet-34 network [HZRS16] followed by six convolutional layers for relative pose prediction. Since relative pose is defined by at least two frames, we enforce the motion estimation network to predict the relative pose between the input frame and its next consecutive frame.

Training details: We use two consecutive frames to train our network in a self-supervised manner. During training, we estimate the dense inverse

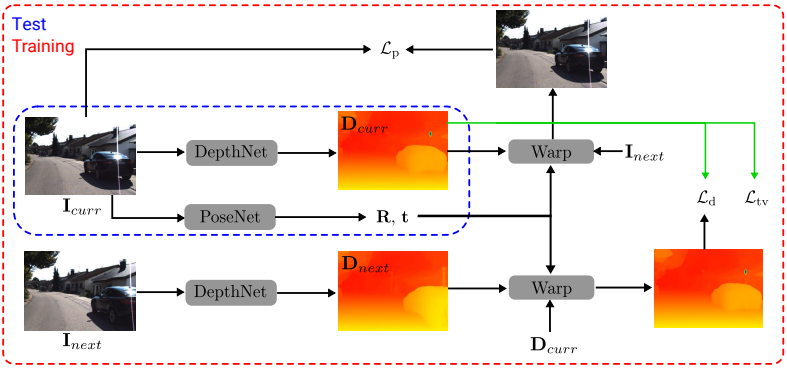


Figure 8.1.: Architecture of single image motion estimation network. Our network consists of a single image depth network and a single image motion estimation network. The depth network predicts the dense inverse depth map $\mathbf{D}_{curr}$ and $\mathbf{D}_{next}$ of $\mathbf{I}_{curr}$ and $\mathbf{I}_{next}$, respectively. We have three losses to train the network. They are a photometric consistency loss $\mathcal{L}_p$, a inverse depth map consistency loss $\mathcal{L}_d$ and a smoothness loss $\mathcal{L}_{tv}$ for the predicted inverse depth map. We use two consecutive frames, i.e., $\mathbf{I}_{curr}$ and $\mathbf{I}_{next}$, for training. During inference, we only use a single image $\mathbf{I}_{curr}$ and the estimated motion is defined as the relative pose from the next frame to current frame.

depth maps $\mathbf{D}_{curr}$ and $\mathbf{D}_{next}$ corresponding to the current frame $\mathbf{I}_{curr}$ and its next consecutive frame $\mathbf{I}_{next}$ respectively by the depth network. We input the current frame $\mathbf{I}_{curr}$ to the pose network, such that it can predict the relative pose $\mathbf{T}_{next}^{curr}$ from $\mathbf{I}_{next}$ to $\mathbf{I}_{curr}$. Since the relative rotation between two consecutive frames is usually small, we parameterize $\mathbf{T}_{next}^{curr}$ by euler angles and a 3-D translation vector.

We adopt the loss functions used in Bian et al. [BLW+19] to train our network in a self-supervised manner. In particular, our loss functions consist of three main parts: a photometric consistency loss, a depth map consistency loss and an edge-aware depth map smoothness loss. More

Method	Seq.09	Seq.10
ORB-SLAM (full)	0.014±0.008	0.012±0.011
ORB-SLAM (short)	0.064±0.141	0.064±0.130
Zhou et al. [ZBSL17]	0.021±0.017	0.020±0.015
Zhou et al.* [ZBSL17]	0.016±0.009	0.013±0.009
Mahjourian et al. [MWA18]	0.013±0.010	0.012±0.011
GeoNet [YS18]	0.012±0.007	0.012±0.009
DF-Net [ZLH18]	0.017 ± 0.007	0.015 ± 0.009
Ranjan et al. [RJB+19]	0.012 ± 0.007	0.012 ± 0.008
Chen et al. [CSS19]	0.011 ± 0.006	0.011 ± 0.009
Ours	0.0296±0.037	0.029±0.043

Table 8.1.: Single image motion estimation results on KITTI odometry dataset [GLU12] with 5-frames absolute trajectory error metric. "$*$" denotes the network is fine tuned after the paper submission. It demonstrates that our network performs better than ORB-SLAM (short) and is competitive compared to other baselines, although our motion estimation network only requires a single image as input.

details can be found from Bian et al. [BLW+19]. The network is trained in 120 epoches with a learning rate of $1e^{-4}$ and a batch size 8.

Evaluation metrics and experimental results: We train and evaluate our networks on KITTI raw dataset [GLU12] using Eigen et al.'s split [EPF14], which is commonly used by the community [ZBSL17, YS18, RJB+19]. We also resize the input images to be 416×128 as prior works. We follow the metrics used by Eigen et al. [EPF14], to evaluate the depth prediction performance of our network against prior works. Both Table 8.2 and Fig. 8.2 present the quantitative and qualitative results of the depth prediction of our network against prior works. It demonstrates that our network performs better than Eigen et al. [EPF14], Liu et al. [LSLR16] and Zhou et al. [ZBSL17], even trained with a single image pose network. Both Eigen et al. [EPF14] and Liu et al. [LSLR16] are trained with ground

truth depths. The network from Zhou et al. [ZBSL17] is similar as ours. However, they use three consecutive frames as the input for the pose estimation network, while ours only require a single frame.

Figure 8.2.: Single image depth prediction of our sinSfM network. **Left:** Input test image. **Right:** Predicted depth map of our trained depth network. It demonstrates that our depth network is able to predict reasonable depth maps, even trained with a single frame motion estimation network.

We also train the same network on the KITTI odometry dataset [GLU12], to evaluate its performance for relative pose estimation. We use the standard training/test data split for our network training and evaluation. In particular, sequence 00-08 are used for training and sequence 09-10 are used for

evaluation. The input image size is resized to 416×128. Since our network can only predict the pose from next frame to current frame, we concatenate the predicted poses of five consecutive frames to compute the 5-frames absolute trajectory error (ATE) metric. This metric is proposed by Zhou et al. [ZBSL17] and commonly used by the community. Table 8.1 presents the quantitative results. It demonstrates that our network performs better than ORB-SLAM (short) and is competitive to other baselines, which usually need at least three frames for the pose estimation network while ours only require a single image.

Figure 8.3.: Sample images from KITTI visual odometry dataset [GLU12]. It is easy for we human beings to predict the camera motions by observing above images only, given the prior knowledge that the images are from KITTI dataset. For example, we can easily predict that the motion corresponding to the top image is turning right and moving forward. Similarly, we can also easily guess the motion of the bottom image is moving straight forward.

Discussions: How is this possible? It is well known that it is impossible to estimate camera motion from a single global shutter image. However, we find that the camera motion from KITTI dataset [GLU12] is very restricted. It mainly has two degree of freedoms motions, i.e., the forward motion and yaw motion. Furthermore, the vehicle is usually driven on and following urban streets, which usually have clear street boundaries. For example, given two sample images from KITTI odometry dataset as shown in Fig. 8.3, it is easy for us to predict that the camera is moving torwards front right for the top image and it is moving straight forward for the bottom image. We therefore hypothesize that KITTI dataset [GLU12] is an "easy" dataset for those learning based approaches, which can already learn many motion cues from semantic information of a single image.

8.3. Upper bound motion cues that a single image motion estimation network can exploit

To verify our hypothesis, we train our single frame motion estimation network with ground truth camera motions. The motivation of this experiment is to investigate the upper bound of the motion cues possessed by a single image from KITTI dataset.

Experimental settings: We use the standard training/test data split of the KITTI odometry dataset [GLU12], to train and evaluate our motion estimation network. The relative pose is parameterized by a 3-D euler angles and a 3-D translation vector. Given the ground truth pose of each frame from the dataset, we can compute the ground truth relative motions to supervise the training of our motion estimation network. The loss function used to train the network is defined as

$$\mathcal{L} = \|\mathbf{t} - \mathbf{t}_{gt}\|_1 + \lambda\|\mathbf{r} - \mathbf{r}_{gt}\|_1, \tag{8.1}$$

where $\mathbf{t}$ and $\mathbf{t}_{gt}$ are the predicted translation vector and ground truth translation vector respectively, $\mathbf{r}$ and $\mathbf{r}_{gt}$ are the predicted euler angles and ground

		Error↓				Accuracy↑		
Methods	Supervision	AbsRel	SqRel	RMS	RMSlog	< 1.25	$< 1.25^2$	$< 1.25^3$
Eigen et al. [EPF14]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al. [LSLR16]	D	0.202	1.614	6.523	0.275	0.678	0.895	0.965
Garg et al. [GKCR16]	D	0.152	1.226	5.849	0.246	0.784	0.921	0.967
Kuznietsov et al. [KSL17]	D+S	0.113	0.741	4.621	0.189	0.862	0.960	0.986
Godard et al. [GMB17]	S	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Zhan et al. [ZGW+18]	S	0.144	1.391	5.869	0.241	0.803	0.928	0.969
Zhou et al. [ZBSL17]	M	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Yang et al. [YWX+18]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian et al. [MWA18]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Wang et al. [WBZL18]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
Yin et al. [YS18]	M	0.155	1.296	5.857	0.233	0.793	0.931	0.973
Zhou et al. [ZUB18]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Ranjan et al. [RJB+19]	M	0.140	1.070	5.326	0.217	0.826	0.941	0.975
Bian et al. [BLW+19]	M	0.137	1.089	5.439	0.217	0.830	0.942	0.975
Chen et al. [CSS19]	M	0.099	0.796	4.743	0.186	0.884	0.955	0.979
Li et al. [LXW+19]	M	0.146	0.927	4.107	0.216	0.819	0.943	0.981
Ours	M	0.205	1.508	6.202	0.273	0.678	0.909	0.967

Table 8.2.: Single image depth estimation results on the test split of KITTI raw dataset [GLU12]. **D** denotes the network is trained with ground truth depth. **S** denotes the network is trained with stereo pairs. **M** denotes the network is trained with a monocular sequence. It demonstrates that our network achieves competitive performance. In particular, it performs better than Eigen et al. [EPF14], Liu et al. [LSLR16] and Zhou et al. [ZBSL17] in terms of the error metrics and accuracy metrics. Note that our network is trained via only two consecutive frames and only requires a single image for the motion estimation network.

truth euler angles, λ is a hyper-parameter and empirically set as 0.01, $\|\mathbf{x}\|_1$ is the $\mathcal{L}_1$ norm of vector $\mathbf{x}$, the units of the euler angles are in degrees. We resize the input images to 416×128 and use a batch size 8. The network is trained in 120 epoches with a learning rate of $1e^{-4}$.

Experimental results: We use the same evaluation metrics as previous section for relative pose estimation. Since our network can only predict the relative pose from next frame to current frame, we concatenate all the poses for the whole sequence to get the trajectories. The 5-frame ATE errors are 0.0282 ± 0.02 and 0.0248 ± 0.0208 for sequence 09 and 10, which is slightly better than that of self-supervised training. Fig. 8.3 presents the estimated trajectories by our single frame motion estimation network against ground truth trajectories. It demonstrates that the network is able to predict the motion direction. However, it struggles to estimate the exact magnitude, which is the same for a human predictor when given only a single image. The experimental results demonstrate that a single image from KITTI dataset [GLU12] already possesses many motion cues to some extent, such that a deep network can predict reasonable motions instead of random motion predictions. It verifies our hypothesis that KITTI dataset [GLU12] is an "easy" dataset for those learning based methods.

8.4. How is the performance of existing networks on dataset with complex motions?

To further verify our hypothesis, we create a large dataset with Carla simulator [DRC+17] to evaluate those learning based approaches. One of the advantages of using synthetic dataset is that we can control the environment (e.g., lighting conditions), such that we can focus on investigating the effect of complex motions to the performance of those networks.

Dataset creation: Since the camera from Carla simulator [DRC+17] is also attached on a ground vehicle, it has the same motion restrictions as that of KITTI dataset [GLU12]. We therefore modify the simulator such

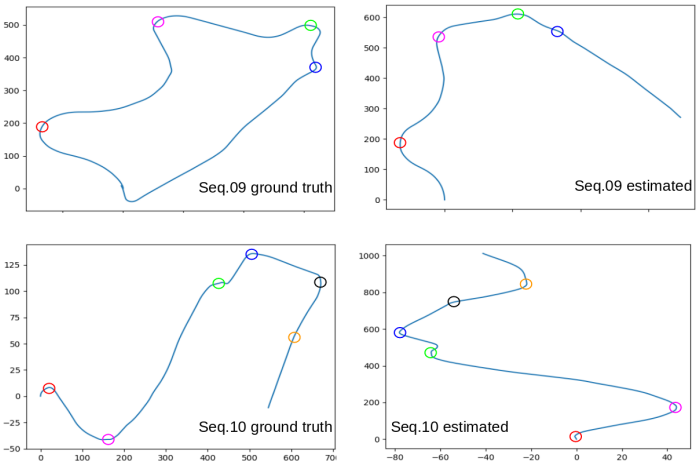


Figure 8.4.: Estimated trajectories against ground truth trajectories on KITTI odometry dataset [GLU12] by our supervised single frame motion estimation network. It demonstrates that our single image motion estimation network is able predict the motion directions for most of the frames. However, it cannot accurately estimate the magnitude, especially for the rotations. It is reasonable since it is also even impossible for we human beings to predict the magnitude accurately given a single image only. It demonstrates that images from KITTI dataset [GLU12] contains many motion cues, such that even a single frame motion estimation network can already learn to predict reasonable motion trajectories instead of random motions. For better visualization, we use colored circles to denote the correspondences.

that the camera can be placed at arbitrary positions and undergoes 6-DoF motions. To have similar settings as that of KITTI dataset, we configure the camera to have a frame rate of 10 Hz. We randomly sample the 3-axis translational velocities of the camera uniformly from 1.5 m/s to 2.5 m/s. Similarly, we randomly sample the 3-axis angular velocities of the camera uniformly from 0 to 0.25 rad/s. We use a constant velocity assumption for each sequence, which contains 20 frames. The resolution of the captured image is 416×256 and we resize it to 416×128 for training and evaluation. We sampled 3885 sequences from the Carla town 01, 02, 03, 05 and 07 with random camera motions. We use the sequences generated from town 01, 02, 03, and 07 for training and the sequences from town 05 for test. In total, we have 3285 sequences (65700 frames) for training and 600 sequences (12000 frames) for test.

Evaluation baselines and metrics: We evaluate one of the best performing state-of-the-art network, i.e., the network from Bian et al. [BLW+19], on our dataset. We use the official implementations from the authors and keep their default hyper-parameters for training. We only change their dataloader such that our dataset can be used for training.

We also implement a simple two-view structure from motion baseline in MATLAB for comparisons. Our baseline adopt the standard pipeline of classic approaches, i.e., feature detection, feature matching, essential matrix estimation, motion decomposition, 3D triangulation, pose and structure joint refinement with bundle adjustment. Since the motion between consecutive frames is usually small for our dataset, we do feature matching with a KLT tracker [BM04].

Since the network from Bian et al. [BLW+19] also predicts the relative pose between two frames, we instead use a histogram of the relative pose errors for performance comparison. Since the translation can only be estimated up to metric scale for all methods, we thus use the angle between the estimated translation vector and the ground truth vector for translation evaluation. For rotation evaluation, we compute the residual rotation matrix between the estimated one and the ground truth rotation. The residual rotation matrix is then converted to euler angle representation and we use these angles for rotation evaluation.

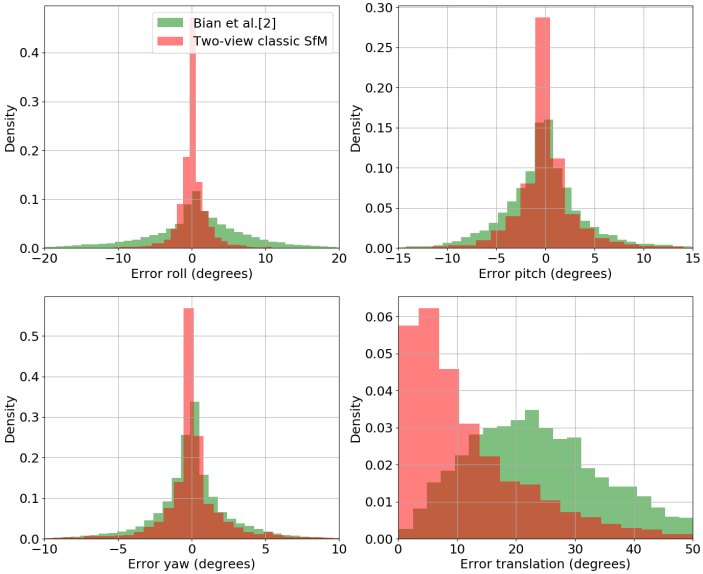


Figure 8.5.: Histograms of the rotation and tranlation errors for both the network of Bian et al. [BLW+19] and our implemented simple two-view classic SfM baseline on the Carla dataset.

Experimental results: Both Fig. 8.5 and Table 8.3 presents the quantitative results for the relative pose estimations. Fig. 8.5 plots the histograms of the relative pose errors. Table 8.3 presents the mean pose errors. Fig. 8.5 demonstrates that most of the translation errors of the network from Bian et al. [BLW+19] concentrate around 21 degrees, while that of the two-view classic SfM baseline concentrates near to 0. We can also find that there are some large translation errors for the two-view classic SfM method. It is caused by the characteristics of our dataset, which usually has small baselines between two consecutive frames. This challenges the method since it usually requires a large parallax between two views for numerically stable solutions.

Method	Roll	Pitch	Yaw	translation
Bian et al. [BLW+19]	0.5192±6.9161	-0.2103±4.3275	0.0267± 2.5892	24.8759±13.3936
Matlab	0.0542±2.3785	0.0063±3.2571	0.0810±2.3610	13.1056±14.2654

Table 8.3.: Relative pose estimation on Carla dataset. All variables have unit in degrees. The network from Bian et al. [BLW+19] is retrained on the Carla dataset for fair comparisons. The experimental results demonstrate that the network from Bian et al. [BLW+19] performs worse than a simple two-view classic SfM pipeline on Carla dataset, which contains complex camera motions.

Discussion: The experimental results demonstrate that those learning based approaches, which work well on KITTI dataset [GLU12], cannot work well on dataset with complex camera motions. The main reason is that KITTI dataset [GLU12] is an "easy" dataset for those existing networks to learn due to its restricted camera motions. Although existing learning based methods achieve remarkable progress on KITTI dataset [GLU12], we argue that there is still much room to improve for general complex camera motions, compared to classic geometric approaches.

8.5. Clarification

Note the main purpose of this work is not to propose a novel network architecture, which is able to do single image VO on KITTI dataset [GLU12]. Instead, we aim to draw the attention of the community that KITTI dataset might not be an ideal dataset for those learning based methods to evaluate on. By demonstrating the fact that even a single image motion estimation network can predict reasonable camera motions, we conclude that the KITTI dataset contain many semantic motion cues for the networks to exploit. We therefore propose the community to use more sophisticated datasets to evaluate learning based methods, such that the trained network can predict camera motions more robustly under various conditions (e.g. for indoor environments where the image does not contain many motion cues).

8.6. Conclusion

In this chapter, we did a study of those unsupervised learning based visual odometry approaches, with an emphasis on the motion estimation network. Our study is driven by the findings that a single image structure from motion is feasible on KITTI dataset [GLU12], and is able to achieve competitive performance with other multiple image based methods. In general, it is impossible to estimate motion from a single image. However, we find that KITTI dataset makes this task possible due to its restricted camera motions.

To further verify our hypothesis, we evaluated one of the best performing state-of-the-art networks [BLW+19] on a simple synthetic dataset with random complex camera motions. We find that those networks cannot work well on this dataset. Although existing works achieve remarkable performance on KITTI dataset [GLU12], we argue that there is still much room for them to improve, even compared to a simple two-view classic structure from motion pipeline, on datasets with general camera motions. Furthermore, more sophisticated datasets instead of only KITTI dataset are also needed for future learning based methods to evaluate on.

9. Conclusion and Outlook

In this chapter, we conclude our dissertation and give a brief discussion on future works.

9.1. Conclusion

In this dissertation, we propose methods to improve the robustness of visual odometry algorithm from three different perspectives, such that it can be reliably deployed to the challenging real world conditions. In particular, we improve the robustness from the hardware perspective, the perspective of deep network enhanced input images, and the algorithmic perspective.

From the hardware perspective, we propose to enlarge the field of view of the pipeline by using a rig of multiple fisheye cameras. Large field of view enables the algorithm to be able to perceive more surrounding environments. The VO pipeline can thus detect more salient feature points for reliable camera motion estimation. It is crucial when the pipeline is operating in challenging complex environments, e.g. at low-lighting conditions and scenarios with many dynamic objects.

Another perspective is to enhance the quality of input images by deep neural networks, which is beneficial for almost all algorithms which require visual inputs with good quality (instead of limiting to visual odometry only). Motion blur and rolling shutter effect are two common factors to degrade the captured images. Motion blur usually occurs when longer exposure time is necessary while the camera is moving (e.g. at low-lighting night conditions). It is well known that many algorithms (e.g. depth prediction, feature detection or object detection) suffer from motion blur. Rolling shutter effect usually occurs while a moving CMOS camera (which is widely integrated on smartphones due to its cheaper price) capturing the

images. In contrast to a global shutter camera, which captures all pixels at the same time, a rolling shutter camera sequentially captures all the pixels row by row. Therefore, different types of distortions, e.g. skew, smear or wobble, will appear if the camera is moving during the image capture. 3D vision algorithms (e.g. structure from motion, visual odometry or depth prediction) suffer from rolling shutter distortions. We therefore propose two deep neural networks to deblur the motion blurred images, and to rectify the input rolling shutter images to global shutter images respectively in this dissertation.

To further improve the robustness of VO method against motion blurred images. We also propose to a hybrid visual odometry algorithm which models the image formation process of motion blur. As conventional algorithms, our method consists of a front-end tracker and a back-end mapper. During tracking, instead of estimating the camera pose at a particular point in time, we estimate the local camera motion trajectory within the exposure time for each frame. It allows us to explicitly model the motion blur in the image and leverage it for tracking. We assume the reference keyframe image is sharp, which is achieved by applying a deep deblurring network on the input motion blurred image. Since keyframes are usually sampled with a frequency much lower than frame-rate, we can thus take advantage of a powerful (less efficient) deep network for keyframe deblurring. The VO algorithm is thus able to compensate the effect of motion blur without deblurring all of them. Experimental results demonstrate that our algorithm improves both the accuracy and robustness over existing methods on both synthetic and real-world datasets.

We also try to study the limitations of existing learning based methods, which aim to improve both the accuracy and robustness of motion estimations by using deep networks. By demonstrating the fact that even a single image motion estimation network can predict reasonable camera motions on KITTI dataset [GLU12], which is commonly used by the community, we conclude that the KITTI dataset might not be sufficient for those learning based approaches to evaluate on. The reason is that the camera motion from KITTI dataset is too restricted. It mainly consists of two degree of freedoms motions, i.e. the forward motion and the yaw motion. The vehicle is also usually driven on and following urban streets, which have clear

street boundaries. It is thus easy for the network to predict the camera motion directions from even a single image, which already contains many semantic motion cues for the network to exploit for motion prediction.

9.2. Future works

This dissertation mainly focuses on improving the robustness of visual odometry algorithms. However, robustness is crucial for almost all vision algorithms, e.g. image based localization, depth estimation and 3D reconstruction. In this section, we will discuss two possible open problems which should raise our attention as future works.

Robust long-term visual localization: Visual localization is a technique used to localize a mobile agent within a pre-built map from captured images. It is a critical component for many robotic applications, e.g. autonomous driving, augmented reality and mixed reality. Despite the community has devoted many efforts to the problem, there still have limitations which limit its practical deployment. For example, one of the limitations is how to robustly localize the camera from a map, which is built under different conditions (e.g. localize an image or an image sequence captured in the spring against a map built in the winter, or captured in the day time against a map built in the night etc.). The state-of-the-art visual localization algorithms usually require to find 2D-3D feature correspondences by matching newly computed 2D feature descriptors against the 2D feature descriptors stored in the map. Apperance changes (e.g. across different seasons, day/night) would affect the robustness of feature matchings significantly, which might further fail the whole localization pipeline. One possible solution is to continuously upadting the map. However, it is not practical for areas which have no mobile agents visiting for a long time. The question is if there will have a more elegant solution? I would vote for yes since we human beings can achieve it easily. Imaging you have visited Big Ben (London) in the last summer day, it would still be easy for you to quickly relocalize yourself even if you re-visit there in the winter night. By incorporating more semantic information (e.g. objects, 3D structural relationship) together with 3D geometry seems to be the way how we human beings achieve this task.

The open problem would thus be how to incorporate semantic information elegantly into existing localization algorithms for robust long-term localization.

Robust 2.5D/3D reconstruction under challenging conditions: Depth estimation and 3D reconstruction play an important role for a mobile agent to perceive its surrounding environment, so that proper decision can be made for human-robot or robot-environment interactions. Most existing methods assume the input images are usually in good conditions. However, it is common that a mobile agent would operate in challenging environments. For example, when the camera tranverses in the rainy weather condition, rain drops would cover the camera lens. Those artefacts affect many stereo matching algorithms, which would result in poor depth estimations. Unreliable 3D scene perception might result in the mobile agent to make false decisions, which would cause tragic accidents (e.g. an autonomous driving car runs out of a highway while carrying a passenger). It is thus cruicial to design algorithms which can robustly estimate dense depth map (i.e. 2.5D reconstruction) and do 3D reconstruction under those challenging conditions.

Bibliography

[ABL16a] H. Alismail, B. Browning, and S. Lucey. Photometric Bundle Adjustment for Vision-Based SLAM. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, 2016. 10

[ABL16b] H. Alismail, B. Browning, and Simon Lucey. Robust tracking in low light and sudden illumination changes. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2016. 4, 51

[AKLP19] Cenek Albl, Zuzana Kukelova, Viktor Larsson, and Tomas Pajdla. Rolling shutter camera absolute pose. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2019. 101

[BBKS10] Simon Baker, Eric Bennett, Sing Bing Kang, and Richard Szeliski. Removing rolling shutter wobble. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 15

[BCC+18a] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 155

[BCC+18b] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM: learning a compact, optimisable representation for dense visual SLAM. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[BLC+19] Michael Bloesch, Tristan Laidlow, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Learning Meshes for Dense Visual SLAM. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 155

[BLW+19] Jia-Wang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised Scale-consistent Depth and Ego-motion Learning from Monocular Video.

In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 155, 156, 157, 158, 159, 163, 166, 167, 168, 170

[BM04] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A Unifying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004. 3, 15, 39, 55, 166

[BNG+16] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *International Journal of Robotics Research (IJRR)*, 2016. 152

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006. 11

[CC18] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 114

[CCC+16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I.D. Reid, and J.J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. In *IEEE Trans. on Robotics*, 2016. 8

[CGG+18] Huaijin Chen, Jinwei Gu, Orazio Gallo, Mingyu Liu, and Jan Kautz. Reblur2deblur: Deblurring videos via self-supervised learning. In *International Conference on Computational Photography (ICCP)*, 2018. 14, 80, 81

[CL09] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM Transactions on Graphics (SIGGRAPH ASIA 2009)*, 28(5), 2009. 13, 71

[CLFP10] Brian Clipp, Jongwoo Lim, Jan-Michael Frahm, and Marc Pollefeys. Parallel, Real-Time Visual SLAM. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2010. 9, 35

[CSS19] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised Learning with Geometric Constraints in Monocular Video: Connecting Flow, Depth, and Camera. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 159, 163

[CY00] James M. Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian

inference. In *Advances in Neural Information Processing Systems (NIPS)*, 2000. 15

[Dav03] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2003. 8, 9

[DC19] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 156

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 72

[DRC+17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. 114, 157, 164

[DRMS07] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera slam. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6), 2007. 8

[DSM11] Tue-Cuong Dong-Si and Anastasios I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2011. 27, 31, 58, 59

[EKC17] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 40(3):611–625, 2017. 1, 2, 3, 4, 6, 9, 10, 11, 27, 53, 56, 57, 58, 127, 128, 130, 131, 142, 144, 145, 147, 151, 153

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 102, 156, 159, 163

[ESC14] Jacob Engel, Thomas Schops, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular slam. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014. 1, 8, 9, 10, 11, 35

[ESC15] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015. 9

[FCDS17] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. on Robotics*, 33(1), 2017. 12

[FPS14] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2014. 3, 6, 9, 10, 11, 35

[FR10] Per Erik Forssen and Erik Ringaby. Rectifying rolling shutter video from hand-held devices. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 15, 106, 107

[FS12] F. Fraundorfer and D. Scaramuzza. Visual odometry: Part ii - matching, robustness, optimization, and applications. *Robotics and Automation Magazine (RAM)*, 19(2), 2012. 8, 35

[FSH+06] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph. In *ACM Trans. on Graphics*, 2006. 13, 71

[FSR+13] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Mhlfellner, Stefan Wonneberger, Julian Timpner, Stephan Rottmann, Bo Li, Bastian Schmidt, Thien Nghia Nguyen, Elena Cardarelli, Stefano Cattani, Stefan Brning, Sven Horstmann, Martin Stellmacher, Holger Mielenz, Kevin Kser, Markus Beermann, Christian Hne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Ren Iser, Rudolph Triebel, Ingmar Posner, Paul Newman, Lars Wolf, Marc Pollefeys, Stefan Brosig, Jan Effertz, Cdric Pradalier, and Roland Siegwart. Toward automated driving in cities using close-to-market sensors, an overview of the v-charge project. In *IEEE Intelligent Vehicle Symposium (IV)*, 2013. 9, 12

[FZG+17] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semi-direct visual odometry for monocular and multi-camera systems. *IEEE Trans. on Robotics*, 33(2), 2017. 9

[GJZ+10] Ankit Gupta, Neel Joshi, C. Lawrence Zitnick, Michael Cohen, and Brian Curless. Single image deblurring using motion density functions. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010. 13, 71

[GKCE12] Matthias Grundmann, Vivek Kwatra, Daniel Castro, and Irfan Essa. Calibration free rolling shutter removal. In *International Conference on Computational Photography (ICCP)*, 2012. 15

[GKCR16] Ravi Garg, B. G. Vijay Kumar, Gustavo Carneiro, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 163

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. ii, iv, 7, 12, 63, 114, 156, 157, 159, 160, 161, 162, 163, 164, 165, 169, 170, 172

[GMB17] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 72, 163

[GOZGJS18] Ruben Gomez-Ojeda, Zichao Zhang, Javier Gonzalez-Jimenez, and Davide Scaramuzza. Learning-based image enhancement for visual odometry in challenging hdr environments. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2018. 51

[GPM+14] Ian-J Goodfellow, Jean PougetAbadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 14

[GTSJ19] Hongyun Gao, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 145

[GWDC18] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2018. 1

[HC16] Lionel Heng and Benjamin Choi. Semi-direct visual odometry for a fisheye-stereo camera. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2016. 44, 46, 47, 48, 49, 61

[HFFR12] Johan Hedborg, Per-Erik Forssn, Michael Felsberg, and Erik Ringaby. Rolling shutter bundle adjustment. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 101

[HHL+15] C. Hane, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweep stereo. In *International Conference on 3D Vision (3DV)*, 2015. 38, 41, 42, 56, 57, 58

[HKZ15] Michal Hradi, Jan Kotera, Pavel Zemk, and Filip roubek. Convolutional neural networks for direct text deblurring. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2015. 14

[HLP13] Lionel Heng, Bo Li, and Marc Pollefeys. CamOdoCal: Automatic Intrinsic and Extrinsic Calibration of a Rig with Multiple Generic Cameras and Odometry. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2013. 9

[HLP14] Lionel Heng, Gim Hee Lee, and Marc Pollefeys. Self-Calibration and Visual SLAM with a Multi-Camera System on a Micro Aerial Vehicle. In *Proc. Robotics: Science and Systems (RSS)*, 2014. 9

[HLP15] Lionel Heng, Gim Hee Lee, , and Marc Pollefeys. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. *Autonomous Robots (AURO)*, 39(3), 2015. 9, 12

[HLW17] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 111

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 111, 112, 157

[IMS+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 73, 74, 110

[ISKB18] E. Ilg, T. Saikia, M. Keuper, and T. Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 83

[JAL16] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 113

[JGR+18] Joel Janai, Fatma Gney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 72, 78

[JSZK15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 79

[KBM+18] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 14, 71, 73, 74, 82, 86, 87

[KF09] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *Advances in Neural Information Processing Systems (NIPS)*, 2009. 13, 71

[KJBL11] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. In *Technical report, Stanford*, 2011. 15

[KL15] Tae Hyun Kim and Kyoung Mu Lee. Generalized video deblurring for dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 13

[KLR17] Jae-Hak Kim, Yasir Latif, and Ian Reid. Rrd-slam: Radial-distorted rolling-shutter direct slam. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017. 101

[KLSH17] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Scholkopt, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 14

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 2, 8, 10, 35, 37

[KMR13] Bryan Klingner, David Martin, and James Roseborough. Street view motion-from-structure-from-motion. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 101

[KMWW19] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 7, 145, 146, 147, 148

[KSL17] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 163

[KUH18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 77

[LAA18] Yizhen Lao and Omar Ait-Aider. A robust method for strong rolling shutter effects correction using lines with automatic feature selection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 15

[LCC08] Chia-Kai Liang, Li-Wen Chang, and Homer H. Chen. Analysis and compensation of rolling shutter effect. In *IEEE Trans. on Image Processing (TIP)*, 2008. 15

[LCLP20] Peidong Liu, Zhaopeng Cui, Viktor Larsson, and Marc Pollefeys. Deep shutter unrolling network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 104, 142

[LFP11] Jongwoo Lim, Jan-Michael Frahm, and Marc Pollefeys. Online Environment Mapping. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 9, 35

[LGH+18] Peidong Liu, Marcel Geppert, Lionel Heng, Torsten Sattler, Andreas Geiger, and Marc Pollefeys. Towards robust visual odometry with a multi-camera system. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2018. 1, 51

[LHSP17] Peidong Liu, Lionel Heng, Torsten Sattler, and Marc Pollefeys. Direct visual odometry for a fisheye-stereo camera. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2017. 1, 9, 36

[LJP+20] Peidong Liu, Joel Janai, Marc Pollefeys, Torsten Sattler, and Andreas Geiger. Self-supervised linear motion deblurring. In *IEEE Robotics and Automation Letters (RA-L)*, 2020. 72

[LKL11] Hee Seok Lee, Junghyun Kwon, and Kyoung Mu Lee. Simultaneous localization, mapping and deblurring. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2011. 5

[LLB+15] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Timothy Furgale. Keyframe-based visualinertial odometry using nonlinear optimization. *International Journal of Robotics Research (IJRR)*, 2015. 12, 27, 35, 58

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 11

[LSK+17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 102, 117

[LSLR16] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2016. 156, 159, 163

[LWDF09] Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman. Understanding and evaluating blind deconvolution algorithm. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 13, 71

[LXW+19] Shunkai Li, Fei Xue, Xin Wang, Zike Yan, and Hongbin Zha. Sequential Adversarial Learning for Self-Supervised Deep Visual Odometry. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 163

[LZVP21] Peidong Liu, Xingxing Zuo, Larsson Viktor, and Marc Pollefeys. Mba-vo: Motion blur aware visual odometry. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 127

[MAMT15] Ral Mur-Artal, J. M. M. Montiel, and Juan D. Tards. ORB-SLAM: A versatile and accurate monocular slam system. *IEEE Trans. on Robotics*, 31(5), 2015. 8, 9, 10

[MAT17a] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. on Robotics*, 33(5):1255–1262, 2017. 2, 3, 4, 6, 9, 127, 142, 145, 147, 150, 151, 153

[MAT17b] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Trans. on Robotics*, 2017. 63, 67, 156

[MHR18] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2018. 72, 78

[MIH+16] Nikolaus Mayer, Eddy Ilg, P. Husser, Philipp Fischer, D. Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 157

[MKA18] Nimisha Thekke Madam, Sunil Kumar, and Rajagopalan A.N. Unsupervised class-specific deblurring. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 14, 82, 86, 87

[MR07a] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, April 2007. 19

[MR07b] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2007. 12, 35

[MvSU+18] Hidenobu Matsuki, Lukas von Stumberg, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. Omnidirectional dso: Direct sparse odometry with fisheye cameras. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3693–3700, 2018. 1

[MWA18] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D Geometric Constraints. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 159, 163

[NKL17] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring.

In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 14, 71, 72, 73, 74, 81, 82, 86, 87, 102, 117

[NNB04] David Nister, Oleg Naroditsky, and James Bergen. Visual Odometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004. 8

[OTFY18] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 155

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 82, 106, 116

[PKD+16] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 72

[PL17] Haesol Park and Kyoung Mu Lee. Joint estimation of camera pose, depth, deblurring and super-resolution from a blurred image sequence. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 13

[PMB+09] Alberto Pretto, Emanuele Menegatti, Maren Bennewitz, Wolfram Burgard, and Enrico Pagello. A visual odometry framework robust to motion blur. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2009. 5, 71

[PMT+17] G. Pascoe, W. Maddern, M. Tanner, P. Pinies, and P. Newman. Nid-slam: Robust monocular slam using normalized information distance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5, 51

[PSP17] S. Park, T. Schops, and Marc Pollefeys. Illumination change robustness in direct visual SLAM. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017. 4, 51

[PZL17] Pulak Purkait, Christopher Zach, and Ales Leonardis. Rolling shutter correction in manhattan world. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 15

[QWMT19] Jiayan Qiu, Xinchao Wang, Stephan J. Maybank, and Dacheng Tao. World from blur. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 71

[RBR17] Vijay Rengarajan, Yogesh Balaji, and A.N. Rajagopalan. Unrolling the shutter: Cnn to correct motion distortions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 16, 101, 113, 114

[RD06] E. Rosten and T. Drummond. Machine learning for high speed corner detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006. 11

[Ric72] W Richardson. Bayesian-based iterative method of image restoration. *Journal of the optical society of America*, 62(1), 1972. 13

[RJB+19] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 159, 163

[RRA16] Vijay Rengarajan, A.N. Rajagopalan, and R. Aravind. From bows to arrows: rolling shutter rectification of urban scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 14, 106

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2011. 11

[SDU+18] David Schubert, Nikolaus Demmel, Vladyslav Usenko, Jorg Stuckler, and Daniel Cremers. Direct sparse odometry with rolling shutter. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1

[SDvS+19] David Schubert, Nikolaus Demmel, Lukas von Stumberg, Vladyslav Usenko, and Daniel Cremers. Rolling-shutter modelling for direct visual-inertial odometry. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2019. 1

[SDW17] Shuochen Su, Mauricio Delbracio, and Jue Wang. Deep video deblurring for hand-held cameras. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 14, 72

[SEE+12] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2012. 142, 143, 145, 149, 150

[SF11] D. Scaramuzza and F. Fraundorfer. Visual odometry: Part i - the first 30 years and fundamentals. *Robotics and Automation Magazine (RAM)*, 18(4), 2011. 8, 35

[SF16] Johannes Lutz Schnberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 121

[SJA08] Q Shan, Jiaya Jia, and Agarwala A. High-quality motion deblurring from a single image. *ACM Trans. on Graphics*, 27(3), 2008. 13, 71, 82

[SKBP13] Olivier Saurer, Kevin Kser, Jean-Yves Bouguet, and Marc Pollefeys. Rolling shutter stereo. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 101

[SMS10] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics (JFR)*, 2010. 27, 58, 59

[SPL16] Olivier Saurer, Marc Pollefeys, and Gim Hee Lee. Sparse to dense 3d reconstruction from rolling shutter images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 101

[SSP19] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 142, 143, 145, 146

[SYLK18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 74, 75, 102, 111

[TGS+18] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 14, 71, 72, 73, 74, 75, 82, 83, 84, 85, 86, 87, 127, 145, 146, 148

[TNPH15] Petri Tanskanen, Tobias Ngeli, Marc Pollefeys, and Otmar Hilliges. Semi-Direct EKF-based Monocular Visual-Inertial Odometry. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015. 11, 35

[TSN+16] Siddharth Tourani, Mittal Sudhanshu, Akhil Nagariya, Visesh Chari, and Madhava Krishna. Rolling shutter and motion blur removal for depth cameras. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2016. 71

[TTB11] Yu-Wing Tai, Ping Tan, and Michael S. Brown. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 33(8):1603–1618, August 2011. 13

[TTLN17a] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[TTLN17b] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 155

[UESC16] Vladyslav Usenko, Jakob Engel, Jorg Stückler, and Daniel Cremers. Direct visual-inertial odometry for stereo cameras. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2016. 12

[UZU+17] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 155

[VMR18] Subeesh Vasu, Mahesh Mohan, and A.N. Rajagopalan. Occlusion aware rolling shutter rectification of 3d scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 16, 101

[WBZL18] Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learng depth from monocular videos using direct methods. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 78, 80, 163

[WHS17] Patrick Wieschollek, Michael Hirsch, and Bernhard Scholkopt. Learning blind motion deblurring. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 14

[WSZP10] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 13

[XJ10] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010. 13, 71

[XRLJ14] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 13

[XWL+19] Fei Xue, Xin Wang, Shunkai Li, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[XZJ13] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 82, 86, 87

[YS18] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 155, 159, 163

[YSWC20] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[YWGC18] Nan Yang, Rui Wang, Xiang Gao, and Daniel Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. In *IEEE Robotics and Automation Letters (RA-L)*, 2018. 143

[YWSC18a] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2

[YWSC18b] Nan Yang, Rui Wang, Jrg Stckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 155

[YWX+18] Zhengheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2018. 163

[ZBLD19] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[ZBSL17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 155, 156, 159, 160, 161, 163

[ZCL17] Bingbing Zhuang, Loong Fah Cheong, and Gim Hee Lee. Rolling shutter aware differential sfm and image rectification. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 15, 103, 115, 116, 119, 120, 121, 122, 123

[ZCS17] Z. Zhang, C.Forster, , and D. Scaramuzza. Active exposure control for robust visual odometry in hdr environments. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017. 4, 51

[ZGW+18] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian M. Reid. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 163

[ZKR+18] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien P. C. Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas A. Funkhouser, and Sean Ryan Fanello. Activestereonet: End-to-end self-supervised learning for active stereo systems. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 72

[ZLH18] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: unsupervised joint learning of depth and flow using cross-task consistency. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 159

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 14

[ZPR+18] Jiawei Zhang, Jinshan Pan, Jimmy Ren, Yibing Song, Linchao Bao, Rynson W.H. Lau, and Ming-Hsuan Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 14

[ZRFS16] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2016. 36

[ZS15] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2015. 9, 11

[ZTJ+19] Bingbing Zhuang, Quoc-Huy Tran, Pan Ji, Loong-Fah Cheong, and Manmohan Chandraker. Learning structure-and-motion-aware rolling shutter correction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 16, 101, 103, 106, 113, 114, 115, 116, 117, 119, 120, 122

[ZUB18] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTAM: Deep Tracking and Mapping. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 163

[ZXJ13] S. Zheng, L. Xu, and J. Jia. Forward motion deblurring. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 13