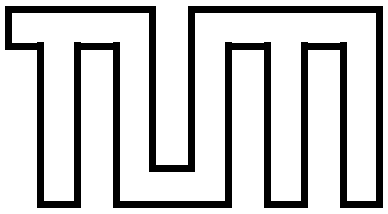
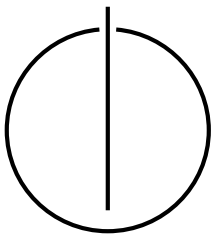


TUM School of Computation, Information and Technology

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Optimization- and Learning-Based Approaches to Visual SLAM and Relocalization

Lukas Michael von Stumberg



TUM School of Computation, Information and Technology

TECHNISCHE UNIVERSITÄT MÜNCHEN

Optimization- and Learning-Based Approaches to Visual SLAM and Relocalization

Lukas Michael von Stumberg

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:	Prof. Angela Dai, Ph.D.
Prüfer der Dissertation:	1. Prof. Dr. Daniel Cremers
	2. Prof. Dr. Davide Scaramuzza
	3. Prof. Maurice Fallon, Ph.D.

Die Dissertation wurde am 24.08.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 19.06.2023 angenommen.

Abstract

Simultaneous Localization and Mapping (SLAM) as well as the related tasks of odometry and relocalization are essential for a number of disruptive technologies, including robotics, autonomous driving and augmented reality.

In the first part of this dissertation, new techniques for optimization-based visual-inertial odometry are explored. Combining cameras and inertial measurement units (IMUs) for pose estimation is a popular and sensible choice, as they are widely-available, low-cost, and complimentary sensors. This thesis proposes novel variants of marginalization to improve consistency of the optimization in cases where estimates change significantly over time. It also introduces a new optimization technique, called pose graph bundle adjustment, which is more accurate than pose graph optimization and faster than bundle adjustment. Based on these techniques, novel strategies for IMU initialization are developed, in which scale and gravity direction are continuously optimized in the main system. The result of these efforts is DM-VIO, an open-source monocular-inertial odometry method. Extensive evaluations on flying drones, handheld, and automotive datasets show that it exceeds the state of the art, including stereo-inertial methods.

In the second part, we investigate how to integrate learned knowledge into visual odometry. Deep neural networks are employed to estimate depths, uncertainty and relative poses. This learned information is combined with an optimization-based odometry system, resulting in a method which outperforms the state of the art in visual odometry by a large margin.

Finally, we address relocalization, in particular the challenge of tracking images across different weather and seasons. In contrast to existing approaches, we employ direct methods and leverage deep neural networks to overcome the limitations which have previously prevented direct methods from being applied in this context. To this end, we devise novel loss formulations, tailored to the requirements of the Levenberg-Marquardt algorithm used in direct image alignment. Further, we employ a pose prediction network to provide an initialization to the optimization. With these techniques, sequences from different seasons are accurately relocalized without relying on feature matching or RANSAC.

Zusammenfassung

Simultaneous Localization and Mapping (SLAM) sowie die damit verbundenen Probleme Odometrie und Relokalisierung sind essentiell für eine Reihe von disruptiven Technologien, darunter Robotik, autonomes Fahren und Augmented Reality.

Im ersten Teil dieser Dissertation werden neue Techniken für optimierungsbasierte visuell-inertiale Odometrie erforscht. Die Kombination von Kameras und inertialen Messeinheiten (IMUs) zur Positionsschätzung ist eine beliebte und sinnvolle Wahl, da es sich dabei um weitverbreitete, kostengünstige und komplementäre Sensoren handelt. Diese Dissertation stellt neue Varianten von Marginalisierung vor, um die Stimmigkeit der Optimierung in Situationen zu verbessern, in denen sich die Schätzung mit der Zeit erheblich ändert. Außerdem wird ein neues Optimierungsverfahren namens Pose Graph Bundle Adjustment eingeführt, welches genauer als Pose Graph Optimization ist und schneller als Bundle Adjustment. Basierend auf diesen Techniken werden neue Strategien zur IMU-Initialisierung entwickelt, bei denen die metrische Größe der Umgebung sowie die Richtung der Gravitation im Hauptsystem kontinuierlich optimiert werden. Das Ergebnis dieser Bestrebungen ist DM-VIO, eine Open Source monokulare visuell-inertiale Odometrie-Methode. Umfangreiche Evaluierungen auf fliegenden Drohnen, handgeführten Kameras und Automobil-Datensätzen zeigen, dass DM-VIO den Stand der Technik übertrifft, einschließlich stereo-inertialer Methoden.

Im zweiten Teil untersuchen wir, wie maschinell gelerntes Wissen für visuelle Odometrie eingesetzt werden kann. Tiefe neuronale Netze werden angewendet, um Tiefenwerte, Unsicherheiten und relative Posen zu schätzen. Diese gelernten Informationen werden mit einem optimierungsbasierten Odometrie-System kombiniert. Das Ergebnis ist eine Methode, die den Stand der Technik in visueller Odometrie bei weitem übertrifft.

Abschließend befassen wir uns mit Relokalisierung, insbesondere mit der Herausforderung, die relative Position von Bildern zu bestimmen, die bei unterschiedlichen Wetterbedingungen und Jahreszeiten aufgenommen wurden. Im Gegensatz zu bestehenden Ansätzen setzen wir dafür direkte Methoden ein und verwenden tiefe neuronale Netze um die Limitationen zu überwinden, die bisher gegen einen Einsatz von direkten Methoden in diesem Kontext gesprochen haben. Dazu konzipieren wir neue Kostenfunktionen, zugeschnitten auf die Anforderungen des Levenberg-Marquardt Algorithmus im Zusammenhang mit Direct Image Alignment. Weiterhin setzen wir ein Netzwerk zur Schätzung von relativen Posen ein, welche als Initialisierung für die Optimierung dienen. Mit diesen Techniken können Sequenzen von unterschiedlichen Jahreszeiten genau relokalisiert werden, ohne dass auf Feature-Matching oder RANSAC zurückgegriffen werden muss.

Acknowledgements

First of all, I would like to thank my doctoral advisor, Prof. Daniel Cremers, for building an awesome research team and providing great support and freedom. I could not have wished for a better advisor!

I would like to thank Prof. Angela Dai, Prof. Davide Scaramuzza, and Prof. Maurice Fallon for agreeing to serve as members of the examination committee. Thanks to Sabine Wagner for ensuring that the group stays organized, and for great help with navigating the bureaucracy. Thanks to Quirin Lohr for his outstanding technical support, going above and beyond to make sure everything keeps running.

I want to thank all my colleagues at the computer vision group for making my time there a great experience. Thanks to Jakob Engel and Jörg Stückler for introducing me to quadrocopters and computer vision, which sparked my desire to pursue a PhD. Thanks to Vladyslav Usenko for collaboration on numerous projects and for his invaluable advice in the beginning of my PhD. The publications achieved during my PhD would not have been possible without my excellent collaborators, I really enjoyed working with all of you: Nan Yang, Patrick Wenzel, Qadeer Khan, Nikolaus Demmel, Felix Wimbauer, David Schubert, Niclas Zeller, Rui Wang, Hidenobu Matsuki, Qing Cheng, Andrej Pangercic.

I want to thank all colleagues during my time at Artisense, it was an exciting ride and I learned a lot.

Lastly but most importantly, I want to thank my spouse and my parents for their incredible support throughout my life, this would not have been possible without you!

Contents

Contents 9

I Introduction and Fundamentals 13

1 Introduction 15
- 1.1 Problem Definition: SLAM, Odometry and Relocalization . . . 16
- 1.2 Sensors . . . 17
- 1.3 Modern SLAM: Between Optimization and Deep Learning . . . 18
  - 1.3.1 Optimization-Based Techniques . . . 18
  - 1.3.2 SLAM and Deep Learning . . . 19
- 1.4 Applications . . . 20

2 Contributions and Outline 23
- 2.1 Visual-Inertial Odometry . . . 26
  - 2.1.1 IMU Initialization . . . 26
  - 2.1.2 Marginalization Procedure . . . 26
  - 2.1.3 Evaluations . . . 27
- 2.2 Improving Visual Odometry with Deep Learning . . . 28
- 2.3 Deep Learning for Direct Relocalization . . . 28

3 Fundamentals 31
- 3.1 Nonlinear Optimization . . . 31
  - 3.1.1 Newton's Method . . . 31
  - 3.1.2 Gauss-Newton Method . . . 32
  - 3.1.3 Levenberg-Marquart Algorithm . . . 33
  - 3.1.4 Robust Huber Norm using Reweighted Least Squares . . . 33
  - 3.1.5 Optimization on a Manifold . . . 34
- 3.2 Optimization and Factor Graphs . . . 35
  - 3.2.1 Factor Graphs . . . 35
  - 3.2.2 Marginalization . . . 36
- 3.3 Optimizations in Direct Sparse Odometry . . . 38
  - 3.3.1 Photometric Bundle Adjustment . . . 38
  - 3.3.2 Marginalization . . . 40
  - 3.3.3 Interactions between Optimizations . . . 41

II Own Publications **43**

4 DM-VIO: Delayed Marginalization Visual-Inertial Odometry **45**
4.1 Introduction . . . 46
4.2 Related Work . . . 48
4.3 Method . . . 49
4.3.1 Notation . . . 49
4.3.2 Direct Visual-Inertial Bundle Adjustment . . . 50
4.3.3 Partial Marginalization using the Schur Complement . . . 51
4.3.4 Delayed Marginalization . . . 52
4.3.5 Pose Graph Bundle Adjustment for IMU Initialization . . . 54
4.3.6 Robust Multi-Stage IMU Initialization . . . 55
4.4 Results . . . 58
4.4.1 EuRoC dataset . . . 58
4.4.2 TUM-VI dataset . . . 60
4.4.3 4Seasons dataset . . . 61
4.5 Conclusion and Future Work . . . 63

5 GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization **65**
5.1 Introduction . . . 66
5.2 Related Work . . . 68
5.3 Deep Direct SLAM . . . 70
5.4 Relocalization Tracking Benchmark . . . 75
5.5 Experimental Evaluation . . . 77
5.5.1 Quantitative multi-weather evaluation . . . 78
5.5.2 Qualitative multi-weather evaluation . . . 81
5.5.3 Additional experiments on EuRoC and CARLA . . . 82
5.6 Conclusion & Future Work . . . 82

6 LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization **85**
6.1 Introduction . . . 86
6.2 Related Work . . . 88
6.3 Method . . . 89
6.3.1 Direct Image Alignment with Levenberg-Marquardt . . . 90
6.3.2 Loss Formulation for Levenberg-Marquardt . . . 91
6.3.3 CorrPoseNet . . . 94
6.4 Experiments . . . 95
6.4.1 CARLA Relocalization Benchmark . . . 98
6.4.2 Oxford RobotCar Relocalization Benchmark . . . 98
6.4.3 Ablation Studies . . . 100

6.4.4 Qualitative Results . . . 100
6.5 Conclusion . . . 100

III Non-Examination-Relevant Own Publications 105

7 Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization 107
7.1 Introduction . . . 108
7.2 Related work . . . 110
7.3 Direct Sparse Visual-Inertial Odometry . . . 111
7.3.1 Notation . . . 111
7.3.2 Photometric Error . . . 112
7.3.3 Inertial Error . . . 112
7.3.4 IMU Initialization and the problem of observability . . . 113
7.3.5 **SIM**(3)-based Representation of the World . . . 113
7.3.6 Scale-aware Visual-inertial Optimization . . . 114
7.3.7 Coarse Visual-Inertial Tracking . . . 120
7.4 Results . . . 121
7.4.1 Robust Quantitative Evaluation . . . 121
7.4.2 Evaluation of the Initialization . . . 122
7.5 Conclusion . . . 126

8 D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry 127
8.1 Introduction . . . 128
8.2 Related Work . . . 131
8.3 Method . . . 132
8.3.1 Self-supervised Network . . . 132
8.3.2 D3VO . . . 136
8.4 Experiments . . . 138
8.4.1 Monocular Depth Estimation . . . 138
8.4.2 Monocular Visual Odometry . . . 141
8.5 Conclusion . . . 145

IV Conclusion and Outlook 147

9 Summary 149

10 Future Research 151

V Appendix **153**

A Multimedia Material **155**

B Open-Source Code and Datasets **159**

C Supplementary Materials **161**

D Original Publications **173**

List of Figures **221**

List of Tables **229**

Own Publications **233**

Bibliography **235**

Part I

Introduction and Fundamentals

Chapter 1

Introduction

"Where am I?" is a fundamental question for not only humans, but for any autonomous agent. Planning, navigation and higher-level scene understanding require a meaningful answer. A distinction can be made, whether the environment is unknown or known in advance. In the former case, the problem is called Simultaneous Localization and Mapping (SLAM). In the latter case, where a prior (SLAM) map is available, the task of relocalization becomes relevant. This dissertation addresses both of these problems. Fig 1.1 shows a live demo of DM-VIO, one of the main contributions of this thesis.

Figure 1.1: DM-VIO (Chapter 4) running live on a laptop, connected to a monocular camera with integrated IMU. It reconstructs the trajectory and 3D environment (pointcloud on the laptop) with remarkable accuracy in real-time. The source-code for this live demo has been released, see Appendix B.

1.1 Problem Definition: SLAM, Odometry and Relocalization

The goal of SLAM is to estimate the trajectory of a sensor system while reconstructing a map of the surrounding environment. A full SLAM system typically comprises the following components:

1. An underlying *odometry* system [14] which is able to recover the ego-motion in unknown environments. Odometry can also be studied on its own [15]–[18], in this case information is forgotten as soon as it leaves the field-of-view. Working on odometry separately can be advantageous, as some datasets (like [19], [20]) do not provide groundtruth across the entire sequence, in which case loop-closed trajectories distort the results. Also, any SLAM system will degenerate to pure odometry, when the operator keeps exploring without revisiting previous places.

2. A *loop-closure* system, which is able to correct the previous trajectory upon revisiting a previous location [21]–[23]. Some systems [24], [25] can also reactivate old keyframes and geometry, enabling to track incoming images against the map.

3. The ability to *relocalize* the agent in the previously generated map. This is closely related to loop-closure and only some SLAM systems have separate capabilities for it [24], [26], [27]. But in contrast to loop-closure, relocalization features a distinct set of problems [28]: It is required to work even under strong appearance and lighting changes which only rarely happen in a single sequence.

In this thesis we first study problem 1, and propose new odometry systems in Chapters 7, 4, and 8. Note that our findings are relevant not only for odometry, but for SLAM in general. All SLAM systems contain an underlying odometry system which can potentially benefit from our improvements.

Afterwards, we develop techniques for tracking images across weathers and seasons in Chapters 5 and 6. This can be applied not only to relocalization (problem 3), but also to long-term loop-closure (problem 2).

1.2 Sensors

The problem of SLAM can be addressed with a variety of sensors. Cameras are an obvious choice as they roughly resemble the way humans perceive the world. They are widely available, low-cost, and passive sensors, which provide rich information. Hence, the field of visual SLAM has become increasingly popular in the last decades [24], [25], [29]–[33].

However, there is another sensor which satisfies these properties: an inertial measurement unit (IMU). IMUs are also low-cost, passive sensors and humans have an analogous sense in the vestibular system. It has been found that cameras and IMUs are complimentary and combining them can overcome their individual disadvantages. Consequently, there has been an emergence of visual-inertial SLAM [34]–[38] and odometry methods [16], [39]–[41] in recent years.

A particular challenge with the combination of the two sensors is the observability of scale. While a monocular camera cannot observe the true scale, an IMU measures metric acceleration. However, it can take an arbitrary amount of time until the scale becomes observable in a visual-inertial system [42] [43], which can be problematic for IMU initialization in such systems. We propose novel solutions to this problem in detail in Chapters 7 and 4, ultimately resulting in a visual-inertial odometry method which outperforms the state of the art.

A sensor combination which can immediately observe the scale is a stereo camera. However, a stereo setup is not always desirable, as it requires sufficient baseline, rigid mounting, and accurate extrinsic calibration. The techniques proposed in Chapters 7, 4, 8 are all monocular, yet they can compete with or even outperform contemporary stereo methods.

Other popular sensors not studied in this dissertation include LiDARs [44] and event cameras [45].

1.3 Modern SLAM: Between Optimization and Deep Learning

1.3.1 Optimization-Based Techniques

The traditional way to address SLAM and visual(-inertial) odometry is through energy minimization. The idea is to define a function determining the quality of the solution, and then to iteratively refine the estimate through means of nonlinear optimization. When it comes to the specific implementation, there is a variety of choices:

- The energy function: The majority of methods minimize the reprojection error [24], [25], [34], [36], [40] , based on a set of predetermined feature correspondences. More recently, a number of direct methods has been proposed [15], [16], [18], [30], [46], [47]. These systems work directly on the image data by minimizing the photometric error. Depending on the implementation, this can result in larger frame-rate [17] or accuracy [15].

- Information density: Methods can be dense [30], [46] (utilizing all image information), semi-dense [18], [32] (using points with sufficient gradient), or sparse [15], [24], [25]. While denser methods utilize more information and result in a more informative reconstruction, sparse methods are usually more efficient. A key advantage of sparse methods is that they better allow for bundle adjustment, as the point observations can be considered independent, resulting in a sparse Hessian matrix.

- Which variables to optimize: There is image alignment [18], [46] which only optimizes the camera poses but fixes geometry; and bundle adjustment [24], [25], which optimizes poses and geometry together. Typically, bundle adjustment is more accurate but encompasses a larger computational burden. Some modern systems like DSO [15] utilize both of them in different stages.

- How to keep the system computationally tractable over time: Most systems only keep a subset of (key-)frames in the active optimization window. Frames leaving this window can either be fixed [24], [48], or marginalized [15], [34], [40].

Despite all these choices / differences, almost all SLAM systems ultimately rely on energy minimization in the core of the estimation process.

We note that there is also a wide range of filtering-based SLAM algorithms [16], [29], [39]. While this is a different paradigm, it is still closely related to optimization-based methods. Disregarding computational differences, a filtering-based SLAM system can be seen as an optimization-based system with just one "iteration", no relinearization, and early marginalization.

1.3.2 SLAM and Deep Learning

In contrast to all techniques summarized in the previous section, some newly emerging techniques based on deep learning follow a completely different paradigm.

Deep learning has taken the computer vision world by storm and impacted almost every field related to it. There are entire areas of computer vision where traditional methods became almost obsolete in a short time frame. The core task behind most computer vision problems is to revert the image formation process, which is a highly ambiguous task. Apparently, learning this based on data is often times superior to handcrafted methods.

However, one of the areas deep learning has not taken over is SLAM. There have been various attempts to learn SLAM from scratch [49]–[53], but they have not been able to compete with the accuracy of traditional methods. The reason for this might be two-fold:

1. There are many parts of a SLAM system which are not ambiguous but can be modelled mathematically. The way points are projected from 3D into the camera image is known in advance and utilizing this knowledge explicitly is advantageous.

2. Optimization-based techniques achieve high accuracy through iterative refinement of the solution, which is hard to beat with neural networks alone.

Utilizing deep learning in SLAM does not mean that one has to relinquish optimization-based techniques. And while key parts of a traditional SLAM system are based on known properties of the image formation process, there are still many handcrafted components. We argue that combining the two techniques is advantageous. The optimization-based foundation ensures that high accuracy is maintained, and injecting learned information into specific parts of the SLAM pipeline can improve robustness and accuracy. Indeed, we observe that works combining deep learning and traditional techniques have shown promising results [54]–[56].

This thesis includes three works which contribute to this emerging field: In Chapter 8, deep networks provide depth, uncertainty, and relative pose information to an optimization-based odometry system. The resulting system clearly outperforms both, traditional and learning based systems, essentially working as well as the state-of-the-art *stereo-inertial* odometry systems, while only using a single monocular camera and no IMU.

In Chapter 5 and Chapter 6, we replace the images in optimization-based direct image alignment with deep features generated by neural networks. Thanks to novel loss formulations, these features improve robustness to bad initializations, and strong lighting and weather changes, which is especially important for relocalization.

These works demonstrate the benefits of combining deep learning techniques with the more traditional optimization-based approaches. We expect that optimization

Figure 1.2: Example of how a quadrocopter can autonomously explore an unknown environment, utilizing a visual SLAM system. This work was published in [1] and is based on the Bachelor's thesis [57] of Lukas von Stumberg.

in SLAM will likely stay relevant, at least as part of hybrid SLAM systems. This also reinforces the importance of the contributions to the more traditional pipeline in the first part of this dissertation.

1.4 Applications

SLAM and relocalization are core parts of many systems, and can be applied in various areas.

Robotics: In order to perform actions and move through a potentially unknown environment, robots need to localize themselves. Hence, SLAM systems are of great importance in robotic systems, especially for moving robots [1], [58], [2]. An example is shown in Fig. 1.2.

Autonomous driving: Autonomous driving is not solved yet, hence it is not fully known, which components will be necessary. However many systems to date utilize HD-maps which provide relevant information to the perception and planning algorithms. Building these maps and localizing the vehicle in them are both core tasks. Especially relocalization across different weather and lighting conditions (Chapters 5, 6) is highly-relevant in this context.

Virtual reality (VR) and augmented reality (AR): VR headsets need to synchronize their motion to the character in the virtual world. This requires accurate poses, which can be obtained using a SLAM system. In AR, a common goal is to present holograms in consistent locations in the world, which also requires precise localization.

Downstream computer vision problems: In addition to the end applications previously mentioned, there are also subsequent computer vision tasks which can benefit from images with accurate camera poses. Examples include multi-view depth estimation [3] (Fig. 1.3) and 3D reconstruction [59].

Figure 1.3: Accurate poses from SLAM can also benefit downstream computer vision tasks. MonoRec [3], a work led by Felix Wimbauer, achieves accurate 3D reconstruction using a cost volume, relying on accurate poses from visual odometry.

Chapter 2

Contributions and Outline

In total, 11 full-length publications and 2 book chapters were published as part of, or in conjunction with this thesis. A summary is presented in Table 2.1.

As the contribution, 3 of these publications [4]–[6] are included in Chapters 4, 5, and 6 respectively. In addition, 2 more full-length publications [7], [8] are included in the non-examination-relevant Chapters 7 and 8.

All of the 5 included publications are full-length papers published in highly-ranked, international, peer-reviewed journals and conferences, including RA-L, ICRA, CVPR, and 3DV. They are the result of joint work with Nan Yang, Patrick Wenzel, Vladyslav Usenko, Qadeer Khan, Rui Wang, and Prof. Daniel Cremers.

These works comprise three main lines of research. First we address visual-inertial odometry and the question of how to optimally combine cameras and IMUs. In particular, we propose novel ways of IMU initialization and improvements to the marginalization procedure. This ultimately results in DM-VIO, an open-source method which outperforms the state-of-the-art in visual-inertial odometry on three datasets.

Afterwards, we incorporate learned knowledge into visual odometry. Depths, uncertainty, and relative poses are predicted by a deep neural network and integrated into an optimization-based odometry system.

Finally, we explore how to track images across weathers and seasons, which is relevant for relocalization and long-term loop closure. Unlike prior work, we approach this problem with direct methods enhanced with deep networks. To combine them effectively, we propose novel loss formulations tailored to the requirements of direct image alignment.

Table 2.1: Chronological list of publications related to this thesis. These include 11 full-length papers published in highly-ranked conferences and journals, as well as 2 book chapters. The works which are not part of this dissertation are shown in gray.

L. von Stumberg, V. Usenko, J. Engel, J. Stueckler, and D. Cremers, **From Monocular SLAM to Autonomous Drone Exploration**, in *European Conference on Mobile Robots (ECMR)*, Sep. 2017. DOI: 10.1109/ECMR.2017.8098709. arXiv: 1609.07835

V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, **Real-Time Trajectory Replanning for MAVs using Uniform B-splines and a 3D Circular Buffer**, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, Sep. 2017. DOI: 10.1109/IROS.2017.8202160. arXiv: 1703.01416

L. von Stumberg, V. Usenko, and D. Cremers, **Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization**, in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2510–2517. DOI: 10.1109/ICRA.2018.8462905. arXiv: 1804.05625 (Chapter 7)

H. Matsuki, L. von Stumberg, V. Usenko, J. Stueckler, and D. Cremers, **Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras**, *IEEE Robotics and Automation Letters (RA-L) & International Conference on Intelligent Robots and Systems (IROS)*, 2018. DOI: 10.1109/LRA.2018.2855443. arXiv: 1808.02775

L. von Stumberg, V. Usenko, and D. Cremers, **A Review and Quantitative Evaluation of Direct Visual–Inertial Odometry**, in M. Yang, B. Rosenhahn, and V. Murino, Eds. Academic Press, 2019, ch. Multimodal Scene Understanding, pp. 159–198, ISBN: 978-0-12-817358-9. DOI: 10.1016/B978-0-12-817358-9.00013-5

D. Schubert, N. Demmel, L. von Stumberg, V. Usenko, and D. Cremers, **Rolling-Shutter Modelling for Visual-Inertial Odometry**, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019. DOI: 10.1109/IROS40897.2019.8968539. arXiv: 1911.01015

L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers, **GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization**, *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 5, no. 2, pp. 890–897, 2020. DOI: 10.1109/LRA.2020.2965031. arXiv: 1904.11932 (Chapter 5)

V. Usenko, L. von Stumberg, J. Stückler, and D. Cremers, **TUM Flyers: Vision—Based MAV Navigation for Systematic Inspection of Structures**, in F. Caccavale, C. Ott, B. Winkler, and Z. Taylor, Eds. Cham: Springer International Publishing, 2020, ch. Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users, pp. 189–209, ISBN: 978-3-030-34507-5. DOI: 10.1007/978-3-030-34507-5_8

N. Yang, L. von Stumberg, R. Wang, and D. Cremers, **D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry**, in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1278–1289. DOI: 10.1109/CVPR42600.2020.00136. arXiv: 2003.01060 (Chapter 8)

P. Wenzel, R. Wang, N. Yang, Q. Cheng, Q. Khan, L. von Stumberg, N. Zeller, and D. Cremers, **4Seasons: A Cross-Season Dataset for Multi-Weather SLAM in Autonomous Driving**, in *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020. DOI: 10.1007/978-3-030-71278-5_29. arXiv: 2009.06364

L. von Stumberg, P. Wenzel, N. Yang, and D. Cremers, **LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization**, in *International Conference on 3D Vision (3DV)*, 2020, pp. 968–977. DOI: 10.1109/3DV50981.2020.00107. arXiv: 2010.06323 (Chapter 6)

F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, **MonoRec: Semi-Supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera**, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. DOI: 10.1109/CVPR46437.2021.00605. arXiv: 2011.11814

L. von Stumberg and D. Cremers, **DM-VIO: Delayed Marginalization Visual-Inertial Odometry**, *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 7, no. 2, pp. 1408–1415, 2022. DOI: 10.1109/LRA.2021.3140129. arXiv: 2201.04114 (Chapter 4)

2.1 Visual-Inertial Odometry

We include two works on visual-inertial odometry: VI-DSO [7] in Chapter 7 (not examination-relevant), and DM-VIO [4] in Chapter 4. Both works combine a photometric energy function with IMU energy terms. The direct back-end with photometric bundle adjustment based on DSO [15] enables the system to track not only corners but any pixels with sufficient gradients. The tight integration of visual and inertial components results in high accuracy and robustness. Other than the combination of vision and IMU, the main contributions of both works can be categorized into IMU initialization, marginalization procedure, and evaluations.

2.1.1 IMU Initialization

IMU initialization is a difficult challenge as it can take arbitrarily long until the scale becomes observable. The previous (published before VI-DSO) best visual-inertial SLAM system [37] waits for 15 seconds before initializing the visual-inertial system and hopes that the scale is observable by this time. In **VI-DSO,** we propose to optimize scale and gravity direction explicitly in the main system. This allows us to initialize immediately with an arbitrary scale, instead of having to wait for the success of a separate IMU initialization. The advantage is that IMU data immediately benefits the robustness of the system, e.g. to fast motions.

However, it also results in the limitation that large-scale outdoor and automotive scenarios can violate the assumptions on the initial scale made by the system. In **DM-VIO** we propose a combination: We do employ a separate, novel multi-stage IMU initializer to make sure that the system works in arbitrarily-scaled environments. But we still optimize scale and gravity direction explicitly in the main system, which ensures that the system works even with a bad initial scale. This allows us to initialize early and still achieve remarkable scale accuracy.

2.1.2 Marginalization Procedure

We employ marginalization of variables to ensure the realtime-capability of the system. The novel initialization strategies devised in VI-DSO and DM-VIO require special treatment of marginalization.

In **VI-DSO** the scale estimate can change a lot over time, which violates a key assumption of marginalization. To overcome this, we propose dynamic marginalization. It maintains a consistent marginalization prior even in cases where a connected variable is far from the optimum initially.

In **DM-VIO**, we strive to answer additional questions which pertain how uncertainties can be transferred between the IMU initializer and the main system. We propose a novel technique called *delayed marginalization*, which provides a solution to these questions. It can also serve as a more flexible alternative to

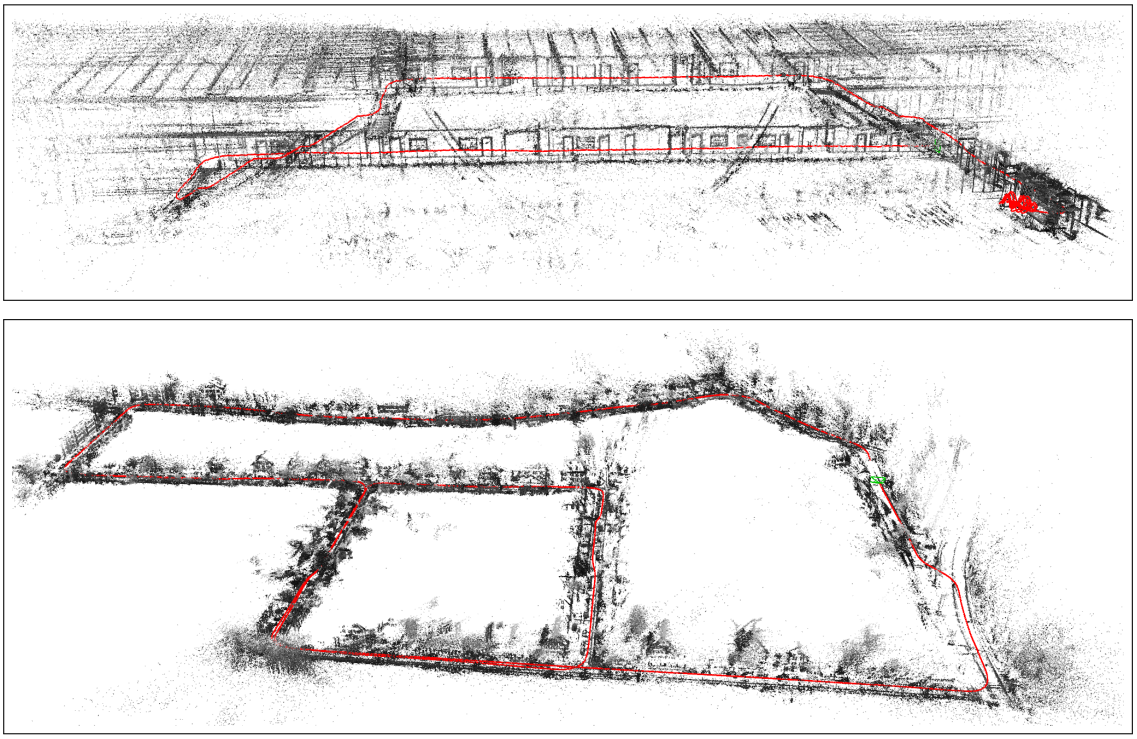


Figure 2.1: Trajectories and pointclouds estimated by DM-VIO. Top: Magistrale of our office building reconstructed using a sequence from the TUM-VI dataset. Bottom: A sequence from the automotive 4Seasons dataset. Long stretches of constant velocity constitute a challenge for monocular visual-inertial odometry, but thanks to our novel IMU initializer, the method works very well.

dynamic marginalization for maintaining a consistent marginalization prior. Lastly, it enables a novel technique called *pose graph bundle adjustment* (PGBA). This is a combination of pose graph optimization and bundle adjustment. In contrast to the former it captures the full visual uncertainty, and in contrast to the latter it does not optimize the points, making it much faster. Delayed marginalization and PGBA are the foundations behind the proposed multi-stage IMU initializer.

2.1.3 Evaluations

VI-DSO is evaluated on the popular EuRoC dataset and we show that it outperforms the previous state of the art.

With **DM-VIO** we not only improve upon these results on EuRoC. We also go one step further and additionally evaluate on the TUM-VI and 4Seasons datasets (see Fig 2.1). In total our evaluations comprise flying drone, large-scale handheld, and automotive scenarios. Across these datasets, DM-VIO outperforms the state of the art in visual-inertial odometry, even compared to popular stereo-inertial methods

while using only a single camera and IMU. Lastly, the full source code for DM-VIO has been released and is available at `https://github.com/lukasvst/dm-vio`.

2.2 Improving Visual Odometry with Deep Learning

In Chapter 8 (not examination-relevant) we include D3VO [8], a work which shows how to integrate learned knowledge into visual odometry. We train a neural network which predicts depth, pose and uncertainty, and then integrate this knowledge into an optimization-based odometry method.

Deep depths: Per-pixel depths are estimated by the network and integrated into the odometry system using the virtual stereo term proposed in Deep Virtual Stereo Odometry (DVSO) [55], but unlike DVSO the network is entirely self-supervised. Different to prior works, we improve self-supervised training using network-predicted affine brightness transformation and photometric uncertainty. This allows our network to outperform state-of-the-art self-supervised depth estimation networks.

Deep pose: Our network estimates the pose between two frames, which we integrate into the odometry system on multiple levels. It is utilized as an initialization and as a prior for both, the frame-to-frame direct image alignment, and for the back-end bundle adjustment. This tight integration of deep poses with odometry significantly boosts the robustness. It resembles the way IMU is integrated in the previous Chapters 7 and 4, but unlike them, this method does not need actual IMU data nor the efforts of time synchronization and calibration associated with it.

Deep uncertainty: As mentioned, the network also estimates the photometric uncertainty for each pixel. The reasoning behind this is that not all surfaces are Lambertian, hence they can violate the brightness constancy assumptions. Downweighting pixels with lower photometric consistency is advantageous, so we feed the predicted uncertainty into the odometry system by replacing the previously handcrafted per-point weight.

Results: We evaluate the proposed odometry system on the Kitti and EuRoC datasets, and show that the integrated deep priors significantly boost robustness and accuracy. The method outperforms traditional monocular visual odometry methods by a large margin. It can even compete with state-of-the-art stereo / LiDAR methods on Kitti, and with state-of-the-art visual-inertial methods on EuRoC.

2.3 Deep Learning for Direct Relocalization

When relocalizing images in an existing SLAM map, significant brightness and weather changes can be present. While direct methods exhibit impressive accuracy

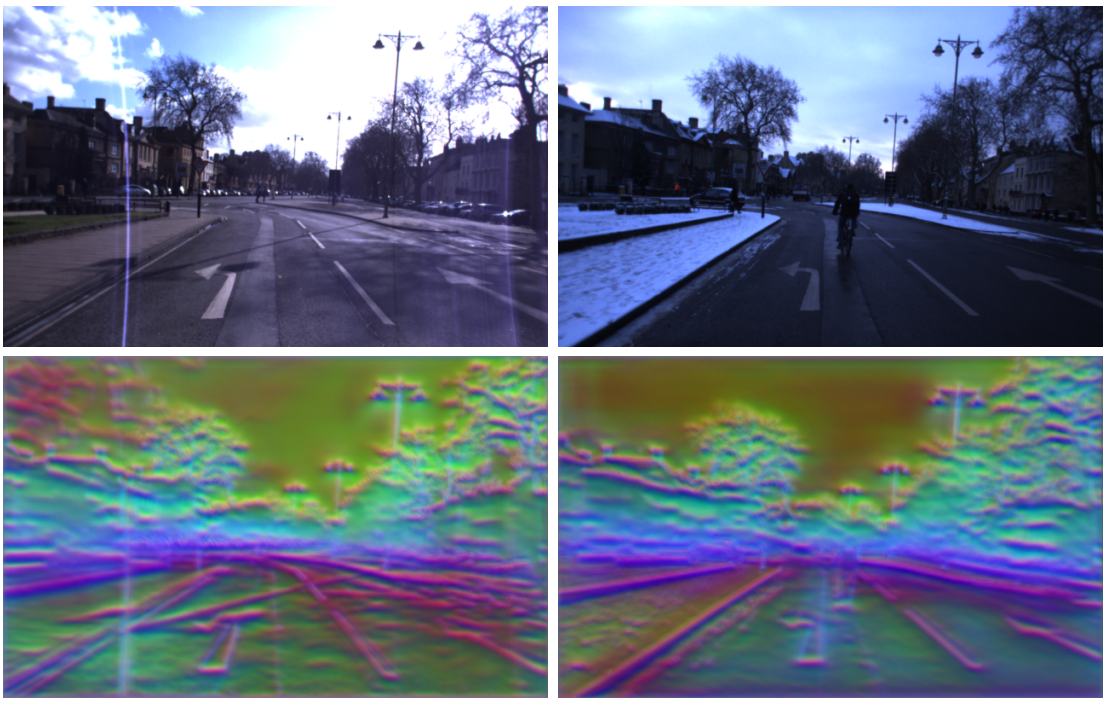


Figure 2.2: Top: Example images from the relocalization tracking benchmark. Bottom: Deep features created by our network LM-Net from these images. Using these features, direct image alignment works well despite strong illumination and weather changes.

and robustness on odometry tasks, they are sensitive to bad initializations and appearance changes, making them unsuitable for relocalization. With our contributions in Chapters 5 (based on [5]) and 6 (based on [6]) we overcome these limitations. This works by replacing the images with deep features trained using groundtruth correspondences.

In Chapter 5 we propose the novel Gauss-Newton loss. It is derived from the Gauss-Newton algorithm for direct image alignment, leveraging the underlying Gaussian probability distribution. During training, the network tries to maximize the estimated probability of the groundtruth correspondence. The loss function allows the network to express certainty in different image directions, which is useful for points lying on lines and other gradients. Using it is superior to relying only on the simpler contrastive loss. Compared to images, the deep features generated by our network GN-Net entail significantly improved robustness against strong lighting changes and bad initializations.

Unless the initialization is almost perfect, the Levenberg-Marquardt (LM) algorithm is usually preferred over the Gauss-Newton algorithm. In Chapter 6 we propose a complete loss formulation and point sampling strategy, tailored to the LM-algorithm. In addition to the Gauss-Newton loss, it consists of three more

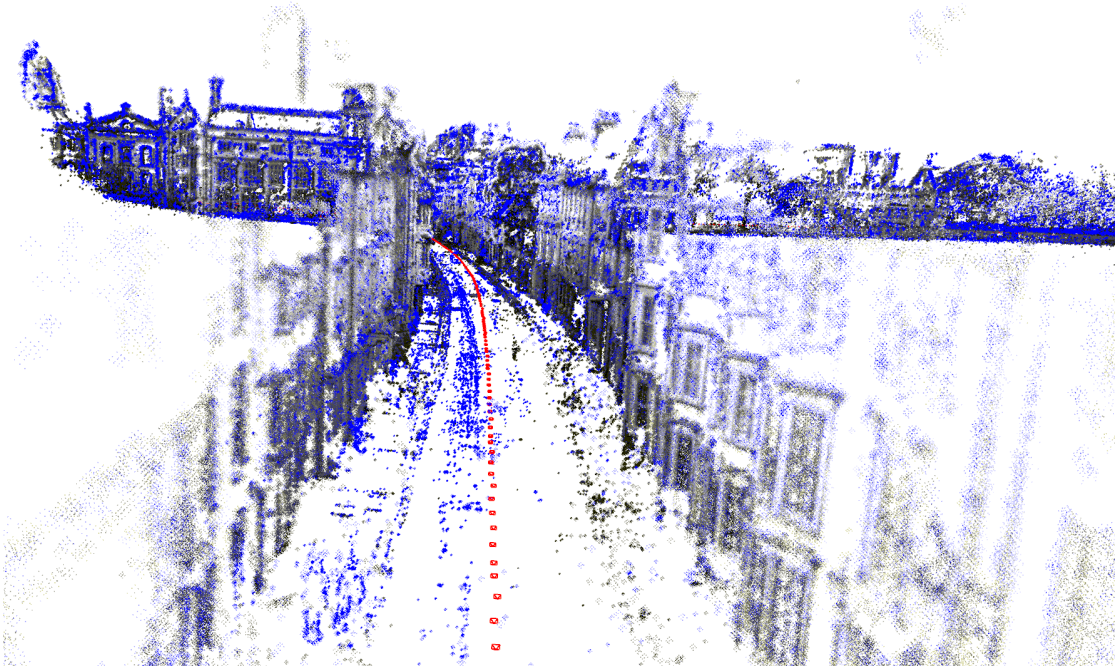


Figure 2.3: Relocalization demo on the Oxford RobotCar dataset. We relocalize sequences with different weather conditions (sunny and snowy in this example) and overlay the pointclouds using the estimated transformations. The pointclouds align very well, showing that the relocalization is accurate.

loss functions derived from the typical behaviour of the LM-algorithm. In our experiments we show that each loss term is critical to achieve best results, verifying our derivations. Consequently, our new network LM-Net significantly outperforms GN-Net.

To further improve the robustness against large baselines, we propose CorrPoseNet in Chapter 6. It is a pose estimation network which provides an initial pose to the direct image alignment. Our full approach called LM-Reloc consists of LM-Net, CorrPoseNet and a nonlinear optimizer.

To evaluate our methods, we create a benchmark for relocalization tracking using sequences from the CARLA simulator and the Oxford RobotCar dataset (Fig. 2.2). In contrast to previous benchmarks, we decouple the tasks of image retrieval and subsequent 6DoF tracking of the localized images, and only focus on the latter in the context of SLAM applications. We verify the effect of the various contributions and show that LM-Reloc is more accurate than indirect methods while being comparable in terms of robustness.

Lastly, we develop a qualitative relocalization demo, demonstrating that the approach can be used for relocalization across weathers in practice (Fig. 2.3).

Chapter 3

Fundamentals

3.1 Nonlinear Optimization

Nonlinear optimization is at the core of the methods developed in this thesis. The idea is usually to devise a nonlinear energy function $E(\mathbf{x})$, which provides an estimate for the quality of a given solution $\mathbf{x}$ to the problem. The solution is then iteratively refined by minimizing the energy function.

In this section we provide a brief overview of the nonlinear optimization techniques applied, for a more complete review we refer the reader to other resources like [60], [61]. The main optimization techniques used throughout this thesis are the Gauss-Newton and Levenberg-Marquardt algorithms. Their derivation is based on Newton's method, which we will start with.

3.1.1 Newton's Method

The foundation of the Newton's method is a second order Taylor expansion of the energy function around the current estimate $\mathbf{x}_0$:

$$E(\mathbf{x}) \approx E(\mathbf{x}_0) + \mathbf{g}^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T \mathbf{H} (\mathbf{x} - \mathbf{x}_0) \tag{3.1}$$

where $\mathbf{g}$ is the Jacobian of the energy function and $\mathbf{H}$ is the Hessian matrix.

$$\mathbf{g} = \frac{dE}{d\mathbf{x}} \text{ and } \mathbf{H} = \frac{d^2 E}{d\mathbf{x}^2} \tag{3.2}$$

The optimal solution to this approximated energy function is at the point, where the derivative of the energy is zero

$$0 \stackrel{!}{=} \frac{dE}{d\mathbf{x}} = g + \mathbf{H}(\mathbf{x} - \mathbf{x}_0) \tag{3.3}$$

Solving this Equation for $\mathbf{x}$ leads to the Newton step, which is used to iteratively refine the solution:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \lambda \left(\mathbf{H}^{-1} \mathbf{g} \right) \tag{3.4}$$

The step size λ is introduced as a parameter to perform more conservative updates, accounting for linearization errors. After each iteration, the energy function is relinearized around the current state estimate, and $\mathbf{H}$ and $\mathbf{b}$ are recomputed.

3.1.2 Gauss-Newton Method

The drawback of Newton's method is that the computation of the Hessian $\mathbf{H}$ can be prohibitively slow. The idea of the Gauss-Newton algorithm is to approximate the Hessian based on the first order derivatives.

For this, the energy function needs to have the following form:

$$E(\mathbf{x}) = \sum_i w_i \cdot r_i(\mathbf{x})^2 \tag{3.5}$$

where, $\mathbf{x}$ is the state vector, r_i are residual functions and w_i is the weight for each residual.

The residuals r_i can be stacked to form the residual vector $\mathbf{r}$ and similarly, the weights can be stacked to a diagonal weight matrix $\mathbf{W}$. Then, the energy can be rewritten as

$$E(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{W} \mathbf{r}(\mathbf{x}) \tag{3.6}$$

We define the Jacobian matrix $\mathbf{J}$ of $\mathbf{r}$ as $\mathbf{J} = \frac{d\mathbf{r}}{d\mathbf{x}}$ and approximate the Hessian with

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{W} \mathbf{J} \tag{3.7}$$

The gradient vector can be computed exactly as $\mathbf{g} = \mathbf{J}^T \mathbf{W} \mathbf{r}$. (Technically, there is an additional factor of 2 in $\mathbf{H}$ and $\mathbf{b}$, but it can be omitted as it cancels out during the update step.)

Throughout this thesis we use a slightly different notation and instead of $\mathbf{g}$, we compute the negative gradient vector

$$\mathbf{b} = -\mathbf{g} = -\mathbf{J}^T \mathbf{W} \mathbf{r} \tag{3.8}$$

resulting in the Gauss-Newton step

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{H}^{-1} \mathbf{b} \tag{3.9}$$

The advantage of the Gauss-Newton method is that the second-order derivatives of the residuals do not need to be computed, yet it can exhibit near-quadratical convergence when the assumptions are met [62]. The approximation of the Hessian (Equation 3.7) is accurate if either the second-order derivative is small, or if the residuals are small, which is often the case for a well-initialized system [61].

3.1.3 Levenberg-Marquart Algorithm

When the system is initialized further from the optimum, the Gauss-Newton algorithm can diverge or converge only slowly. In these cases, the Levenberg-Marquardt algorithm is advantageous. It operates with a damped Hessian, resulting in smaller updates and more robustness.

The damped Hessian can be computed with Levenberg's formula [63]

$$\mathbf{H}' = \mathbf{H} + \lambda \mathbf{I} \tag{3.10}$$

or Marquardt's formula [64]

$$\mathbf{H}' = \mathbf{H} + \lambda \operatorname{diag}(\mathbf{H}) \tag{3.11}$$

Using $\mathbf{H}'$ (from either Equation) instead of $\mathbf{H}$ in the update step (Equation 3.9) effectively works as a combination of the Gauss-Newton method (small λ) and gradient descent (large λ). During optimization, the parameter λ is modified, depending on the success of the current iteration. If the error was reduced, λ is decreased. If the error was not be reduced, λ is increased, resulting in a smaller, more conservative step size.

3.1.4 Robust Huber Norm using Reweighted Least Squares

An underlying assumption of least squares optimization is that measurement errors follow a Gaussian distribution. Especially for image measurements this can be violated, making the squared norm used in the error function (Equation 3.5) suboptimal.

The main problem with the squared norm is that it greatly accentuates outliers, which can result in poor robustness. Hence, robust norms like the Huber norm are often utilized instead. The Huber norm is a combination of a squared norm for small residuals and a linear function for large residuals. It provides more resilience to outliers, while still being a convex function. The Huber norm γ for a residual r can be computed as

$$\left\| r \right\|_{\gamma} = \begin{cases} \frac{r^2}{2}, & \text{for } |r| \leq k \\ k\left(|r| - \frac{k}{2}\right), & \text{for } |r| > k \end{cases} \tag{3.12}$$

where the parameter k defines the maximum residual which is still computed with the squared norm.

Note that an energy with a robust norm does not strictly have the least-squares form (like Equation 3.5) required by the Gauss-Newton algorithm. In order to still apply it, robust norms are usually implemented using iteratively reweighted least squares. This works by rewriting the energy function in a least-squares form with a residual-dependent weight which is chosen to result in the same update step

as the original energy with the robust norm. The weight is recomputed for each iteration, but considered fixed during the computation of the update step. For details, we refer the reader to [65]. This technique is an approximation as it ignores the derivative of the weight during the iteration, but in practice it works very well.

Whenever the Huber norm is used in this dissertation, it is implemented using iteratively reweighted least squares.

3.1.5 Optimization on a Manifold

In the previous sections, the optimized variables are represented using a state vector $\mathbf{x}$. But in SLAM, one typically works with 6DoF poses, consisting of rotation and translation. For usage in a nonlinear optimization framework, a minimal representation is necessary. Throughout this dissertation we employ Lie Groups for this purpose [60], [66]. Rotations are represented using $\mathbf{SO}(3)$, poses using $\mathbf{SE}(3)$, and scaled transformations with $\mathbf{SIM}(3)$. The following explanations operate on $\mathbf{SE}(3)$, but they work equivalently for the other Lie groups.

A vector $\boldsymbol{\xi} \in \mathbb{R}^6$ can be mapped to the corresponding twist (Lie algebra element) $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ using the hat operator $\wedge$. The reverse operator $\vee$ maps from twist to vector. In some literature vectors and Lie algebra elements are used interchangeably. A Lie algebra element can be converted to the Lie group element $\mathbf{T} \in \mathbf{SE}(3)$ and back using the exponential and logarithm map:

$$T = \exp\left(\hat{\boldsymbol{\xi}}\right) \text{ and } \boldsymbol{\xi} = \log\left(\mathbf{T}\right)^{\vee} \tag{3.13}$$

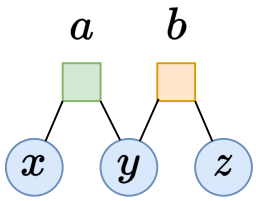
During optimization, the tangent space around a previous solution $\mathbf{T}_0$ is built. Jacobians are computed with respect to increments to this solution. In the following example, the energy only depends on a single pose. The Jacobian can be computed as

$$\mathbf{J} = \frac{d\mathbf{r}(\mathbf{T}_0 \exp\left(\hat{\boldsymbol{\xi}}\right))}{d\boldsymbol{\xi}} \tag{3.14}$$

and the update step (Equation 3.9) is modified to

$$\mathbf{T}_{t+1} = \mathbf{T}_t \exp(\hat{\boldsymbol{\delta}}) \text{ with } \boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b} \tag{3.15}$$

For energy functions depending on a combination of Lie algebra and vector-valued variables, this update is used for the former and a normal addition for the latter. In this example, the increment was placed on the right side of the pose, but it can also be multiplied from the left. The two formulations are equivalent, but the respective variant needs to be applied consistently throughout Jacobian and update computation.



$$E(x, y, z) = a(x, y) + b(y, z)$$

Figure 3.1: Simple factor graph which visualizes the probability density shown in Equation 3.16, and also the corresponding energy function given in Equation 3.17. There is a circle for each variable. Each summand of the energy function is represented by a square, connected to the variables, it depends on.

3.2 Optimization and Factor Graphs

In this section, we discuss the notion of factor graphs, which are used to graphically display optimization systems and their variables. We also provide an explanation of marginalization and its effects on the factor graph.

3.2.1 Factor Graphs

Factor graphs are bipartite graphs, which show how a composite probability function can be divided into its factors [67]. For example consider the following probability distribution:

$$f(x, y, z) = f_A(x, y) f_B(y, z) \tag{3.16}$$

The factor graph for this distribution is shown in Fig. 3.1. It contains one square for each factor of the probability density and a circle for each variable. Lines connect the factors with the variables they depend on.

In this thesis we mostly work with energy functions. Probabilities and energy functions are closely related. When trying to find the solution for x, y, z with maximal probability according to f, one can minimize the negative log-likelihood

$$E(x, y, z) = -ln(f(x, y, z)) = a(x, y) + b(y, z) \tag{3.17}$$

with $a(x, y) = -ln(f_A(x, y))$ and $b(y, z) = -ln(f_B(y, z))$. Hence, each *factor* in the factor graph represents a *summand* of the energy function.

Factor graphs are useful to visualize how the different summands and variables of an energy function interact, and to describe the sparsity structure of the specific problem. Additionally, GTSAM [68] can help to optimize an energy function once it has been described as a factor graph.

3.2.2 Marginalization

Probabilistic interpretation

When exploring the environment for some time, more and more information can be accumulated. In this case, optimization of all variables will become prohibitively slow in a realtime setting. Instead, a sliding window system is preferred. The goal is to remove old variables from the active optimization window, while preserving as much information as possible.

In a probabilistic setting, the optimal solution to this is *marginalization*. Given a joint probability distribution $p(a, b)$ with two variables a and b, marginalization of b will result in (see [62, p. 68])

$$p(a) = \int_b p(a, b) \tag{3.18}$$

For Gaussian probabilities represented with mean and covariance, marginalization is very simple and can be achieved by just taking the corresponding subblock of the matrices [62, p. 68]. Given a partitioning of the variables into the sets α and β, the Gaussian can be represented as

$$\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_\alpha \\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\alpha\alpha} & \boldsymbol{\Sigma}_{\alpha\beta} \\ \boldsymbol{\Sigma}_{\beta\alpha} & \boldsymbol{\Sigma}_{\beta\beta}\end{bmatrix}\right) \tag{3.19}$$

The distribution resulting from marginalization of the variables in β is simply

$$\mathcal{N}\left(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_{\alpha\alpha}\right)) \tag{3.20}$$

Marginalization in the Gauss-Newton algorithm

In the Gauss-Newton algorithm used throughout this dissertation, the multivariate Gaussian is not represented with mean and covariance, but with Hessian $\mathbf{H}$ and gradient vector $\mathbf{b}$. The Hessian is equal to the inverse covariance matrix $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$. Using this, it can be shown that marginalization with this representation can be performed using the Schur complement. With the same partitioning into α and β as above, $\mathbf{H}$ and $\mathbf{b}$ are written as

$$\mathbf{H} = \begin{bmatrix}\mathbf{H}_{\alpha\alpha} & \mathbf{H}_{\alpha\beta} \\ \mathbf{H}_{\beta\alpha} & \mathbf{H}_{\beta\beta}\end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix}\mathbf{b}_\alpha \\ \mathbf{b}_\beta\end{bmatrix} \tag{3.21}$$

We can marginalize the variables in β with the Schur complement:

$$\widehat{\mathbf{H}_{\alpha\alpha}} = \mathbf{H}_{\alpha\alpha} - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\mathbf{H}_{\beta\alpha} \tag{3.22}$$

$$\widehat{\boldsymbol{b}_\alpha} = \boldsymbol{b}_\alpha - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\boldsymbol{b}_\beta \tag{3.23}$$

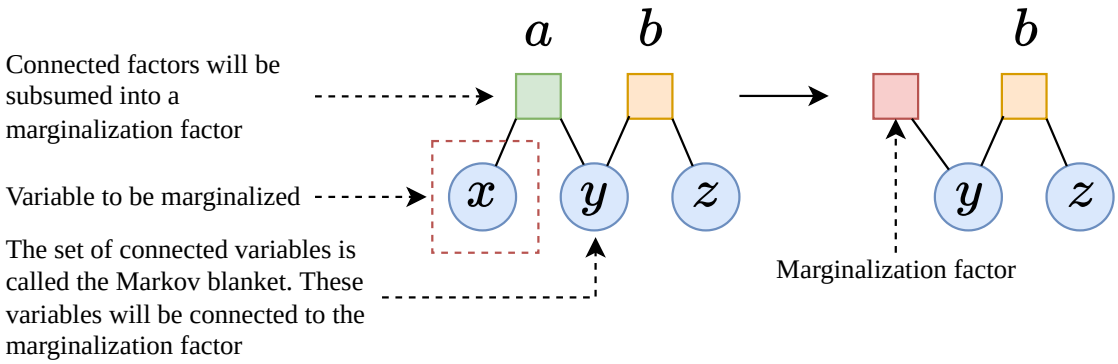


Figure 3.2: Factor graphs before and after marginalization of variable x. The marginalization factor replaces all connected factors and is connected to the variables which form the Markov blanket of x.

This computation is more computation-intensive than marginalization with given mean and covariance. Note however, that only the block corresponding to the marginalized variables $\mathbf{H}_{\beta\beta}$ needs to be inverted. This makes the computation efficient if the number of marginalized variables is small, or if $\mathbf{H}_{\beta\beta}$ is (block)-diagonal. As $\mathbf{H}_{\beta\beta}$ is not always invertible, in practice the Moore-Penrose inverse is often used instead of the normal inverse in Equation 3.22.

Apart from the above derivation using probability theory, we also want to give an intuition for marginalization, based on the Gauss-Newton step. In the original system, the step will be computed with Equation 3.9 as

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_\alpha \\ \boldsymbol{\delta}_\beta \end{bmatrix} = \mathbf{H}^{-1}\mathbf{b} \tag{3.24}$$

We can instead compute the update with the system after marginalization (Equation 3.22):

$$\widehat{\boldsymbol{\delta}_a} = \widehat{\mathbf{H}_{\alpha\alpha}}^{-1}\widehat{\boldsymbol{b}_\alpha} \tag{3.25}$$

It can easily be shown that $\boldsymbol{\delta}_a = \widehat{\boldsymbol{\delta}_a}$, which results from the fact, that Schur complement resembles elimination of variables from the linear equation system. This demonstrates the power of marginalization: It is equivalent to solving the potentially much larger and slower system. The drawbacks are that the marginalized variables cannot be relinearized, and that the update for the marginalized variables is "hidden".

Marginalization and factor graphs

In a larger optimization system represented as a factor graph, marginalization does not need to involve all variables and factors.

Consider the factor graph shown in Fig 3.2 (left), where x shall be marginalized. This will impact all factors connected to x (in this case a). The variables connected to these connected factors (in this case y) form the Markov blanket. These factors will be linearized to obtain the system in Equation 3.21. In our example, x corresponds to β, and y to α. Variable z and factor b are not involved in the marginalization process.

The result of marginalization is a marginalization factor which is connected to all variables in the Markov blanket (in this case y). It is represented by $\widehat{\mathbf{H}_{\alpha\alpha}}$ and $\widehat{\boldsymbol{b}_{\alpha}}$ obtained with Equations 3.22 and 3.23.

After marginalization, the removed factors cannot be relinearized any more. In many cases it is also necessary to fix the linearization points of variables connected to a marginalization factor (y in this example). This technique is called First-Estimates Jacobians [69] and is employed to preserve the nullspaces of the system. Because of this, all variables connected to a marginalization factor should already be close to the correct solution and not change significantly afterwards.

3.3 Optimizations in Direct Sparse Odometry

As an example of how optimization works in a modern SLAM / odometry system, we examine Direct Sparse Odometry [15] in this section. In particular, we focus on the interaction between the different optimization procedures and on the optimization. For further details, we refer the reader to the original publication [15].

3.3.1 Photometric Bundle Adjustment

The main optimization performed in DSO, is a bundle-adjustment-like operation. This means, that camera poses and 3D geometry are optimized together, instead of alternating them like in [18]. This joint optimization is key to achieve accurate results.

The system optimizes a photometric energy function. In contrast to an indirect formulation based on feature correspondences like many other systems, this energy allows to include any point with sufficient gradient in the optimization.

Geometry is represented as a pointcloud. Points are not expressed as 3D coordinates, but are hosted in a keyframe. Each point is defined by the pixel coordinate in its host frame. During optimization the inverse depth of each point is estimated. The 3D coordinate of a point can then be computed by projecting the point into 3D using the estimated point depth, as well as the pose of the host frame.

For computing the photometric energy for a point $\mathbf{p}$, it is projected into all other keyframes j and the brightness values of a patch $\mathcal{N}_{\mathbf{p}}$ around the point are

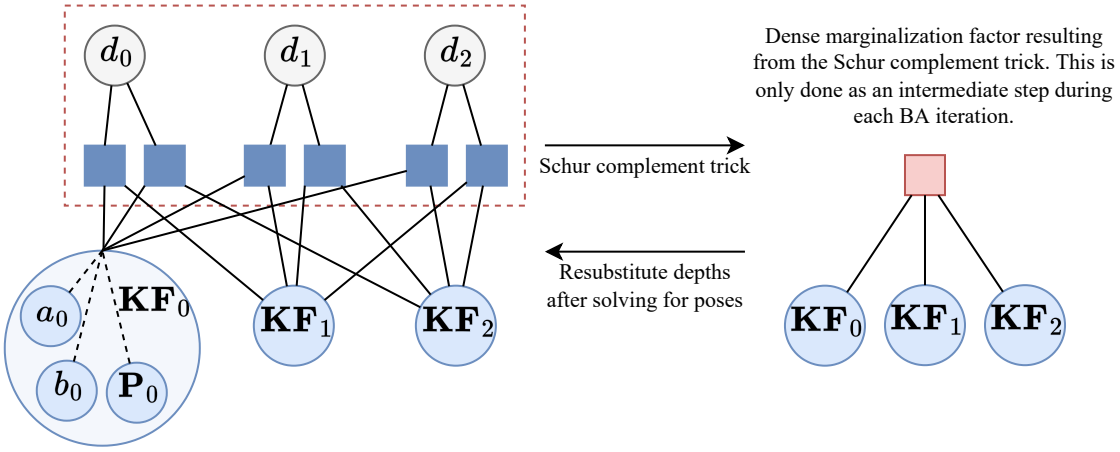


Figure 3.3: *Left:* Factor graph for the bundle adjustment performed in DSO. $\mathbf{P}_i$ is the pose of keyframe i, and d_i is the inverse depth of point i. In this example, there are three keyframes with one point each (d_i is hosted in $\mathbf{KF}_i$ respectively). Each residual depends on the host keyframe, the inverse depth of the point and the target keyframe which the point is projected into. *Right:* System after the Schur-complement trick used to speed up computation of the Gauss-Newton step. It works by "marginalizing" all depth values, which is very efficient, because the depths values are independent. After solving the system on the right for poses, the update for depth values is resubstituted.

compared to the host keyframe using the following function:

$$E_{\mathbf{p}j} = \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} \omega_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma \tag{3.26}$$

where I_i is the host image, I_j is the other image, $\mathbf{p}'$ is the point projected into keyframe j, $\omega_\mathbf{p}$ is the point weight, and t_i and t_j are the exposure times of the images. To account for unmodelled brightness differences and for cases of unknown exposures, an affine brightness change between the images is modelled with the estimated parameters a and b. γ denotes the Huber norm.

A factor graph for the entire optimization is shown in Fig. 3.3 (left). Points are projected into all non-host keyframes, resulting in a residual each. Residuals depend on the inverse depth of the point, the host keyframe, and the target keyframe.

Computation of the Gauss-Newton step (Equation 3.9) involves inverting the Hessian. In DSO, there are over 1000 points, making it intractable to compute the Hessian explicitly. Instead, the Schur-complement trick is used. The idea is to first "marginalize" all landmarks. This marginalization using the Schur complement is very efficient, because the points are independent of each other (there is no residual depending on two or more inverse depths). This means that the matrix $\mathbf{H}_{\beta\beta}$ in Equation 3.22 is fully diagonal and its inversion is trivial. Note that there is no

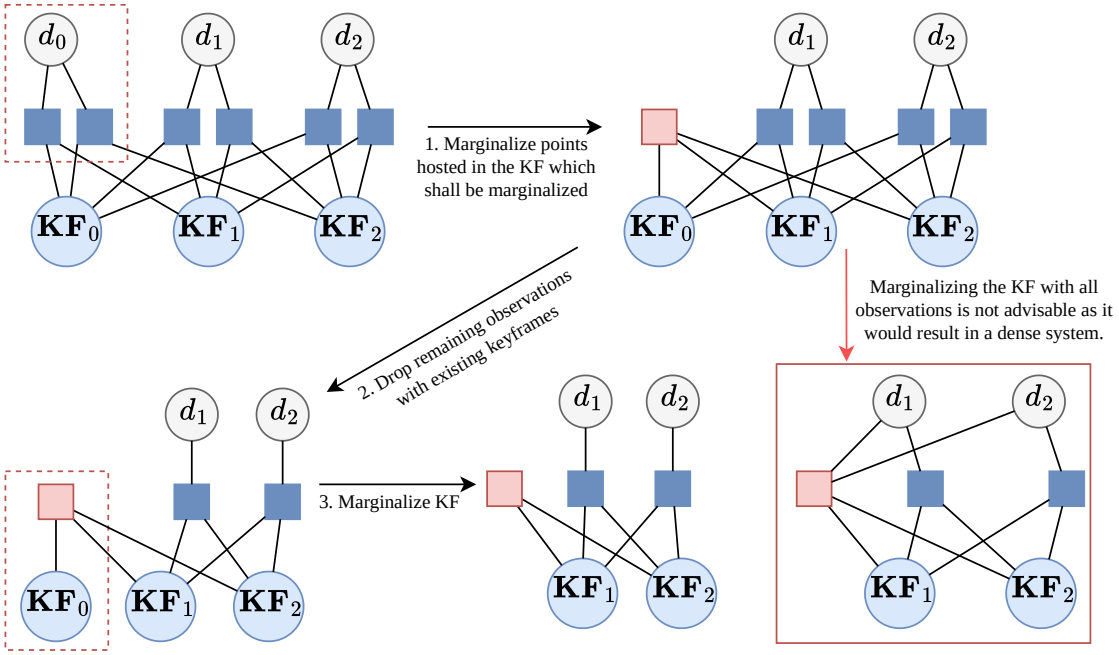


Figure 3.4: Marginalization of $\mathbf{KF}_0$. This involves multiple steps in order to retain the sparsity of the system.

other matrix inversion involved in the Schur complement.

The result of the procedure is the factor graph shown in Fig. 3.3 (right). The Hessian for this factor graph is much smaller than the full system on the left. Using it, the update for the poses (and affine brightness parameters) is computed. Afterwards, the full system is resubstituted again, to compute the update for the inverse depth values. This means, that this procedure is not really a marginalization in the traditional sense, as it only serves as an intermediate step in each bundle adjustment iteration.

3.3.2 Marginalization

To facilitate realtime capability, the bundle adjustment is performed for a maximum of 8 active keyframes, which is achieved using marginalization.

Fig. 3.4 shows the procedure for marginalization of a keyframe. First, all points hosted in the keyframe are marginalized. Then, all observations of active points hosted in other keyframes with the to-be-marginalized keyframe are dropped, and finally the keyframe itself is marginalized. Dropping observations is necessary to retain the sparsity of the Hessian: Skipping this step would result in the factor graph in the bottom right (with red frame), where the depths are connected to the marginalization factor. This would prevent the Schur complement trick shown in Fig. 3.3.

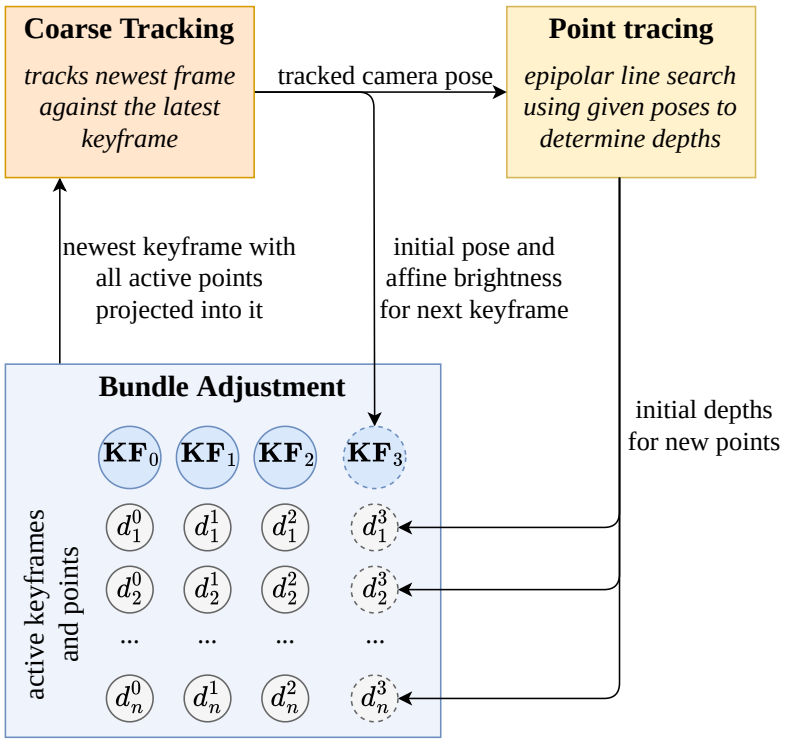


Figure 3.5: Interactions between the different optimizations present in DSO. While the *bundle adjustment* achieves high accuracy, it requires a good initialization and is only performed for keyframes. The *coarse tracking* estimates the pose of each frame using 2-frame direct image alignment against the latest keyframe utilizing the accurate depths from the bundle adjustment. Using the pose estimate from the coarse tracking, *point tracing* performs epipolar line search for a set of candidate points to obtain a rough depth value. When a new keyframe shall be added to the bundle adjustment, its pose is initialized with the result from coarse tracking, and depth values for new points are initialized with the result from point tracing.

One thing to highlight is that the keyframe marginalized is not always the oldest one. Instead, there is a marginalization strategy which tries to keep a combination of old and newer keyframes. This was shown to be superior to a more simple fixed-lag smoother, as points can be observed for a longer period of time.

3.3.3 Interactions between Optimizations

A relevant property of the direct error formulation is that the image gradients used during optimization are only valid in a small vicinity around the correct solution. This implies that a good initialization is critical, especially for the bundle adjustment. To ensure this, DSO uses a combination of different optimizations. Their interactions are shown in Fig. 3.5.

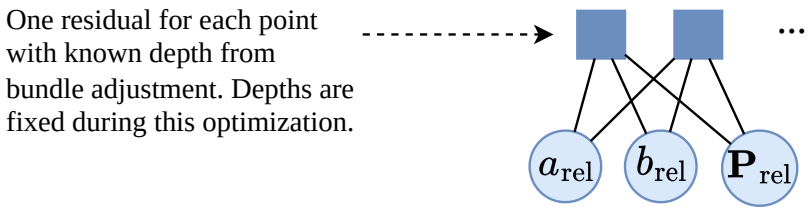


Figure 3.6: Factor graph for the coarse tracking (2-frame direct image alignment).

There are two things to highlight. Firstly, all variables added to the bundle adjustment (poses, affine brightness, and inverse depths) receive an initial value from either the coarse tracking or point tracing. Secondly, the depth values obtained from the point tracing are not used anywhere in the system, except as an initialization to the bundle adjustment. The coarse tracking only relies on accurate, bundle-adjusted depth values.

This is necessary, as the coarse tracking fixes the all point depths during optimization. It only estimates the relative pose and relative affine brightness change between the latest keyframe and the current frame (Fig. 3.6). By optimizing only 8 parameters in total, the optimization is quite fast. To improve robustness, a pyramid scheme is applied.

Part II

Own Publications

Chapter 4

DM-VIO: Delayed Marginalization Visual-Inertial Odometry

Authors	Lukas von Stumberg[1]	stumberg@in.tum.de
	Daniel Cremers[1]	cremers@in.tum.de
	[1]Technical University of Munich, Munich, Germany	

Publication: L. von Stumberg and D. Cremers, **DM-VIO: Delayed Marginalization Visual-Inertial Odometry**, *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 7, no. 2, pp. 1408–1415, 2022. DOI: 10.1109/LRA.2021.3140129. arXiv: 2201.04114

Contribution		
	Problem definition	*significantly contributed*
	Literature survey	*significantly contributed*
	Algorithm development	*significantly contributed*
	Method implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

Abstract We present DM-VIO, a monocular visual-inertial odometry system based on two novel techniques called delayed marginalization and pose graph bundle adjustment. DM-VIO performs photometric bundle adjustment with a dynamic weight for visual residuals. We adopt marginalization, which is a popular strategy to keep the update time constrained, but it cannot easily be reversed, and linearization points of connected variables have to be fixed. To overcome this we propose delayed marginalization: The idea is to maintain a second factor graph, where marginalization is delayed. This allows us to later readvance this delayed graph, yielding an updated marginalization prior with new and consistent linearization points. In addition, delayed marginalization enables us to inject IMU information into already marginalized states. This is the foundation of the proposed pose graph bundle adjustment, which we use for IMU initialization. In contrast to prior works on IMU initialization, it is able to capture the full photometric uncertainty, improving the scale estimation. In order to cope with initially unobservable scale, we continue to optimize scale and gravity direction in the main system after IMU initialization is complete. We evaluate our system on the EuRoC, TUM-VI, and 4Seasons datasets, which comprise flying drone, large-scale handheld, and automotive scenarios. Thanks to the proposed IMU initialization, our system exceeds the state of the art in visual-inertial odometry, even outperforming stereo-inertial methods while using only a single camera and IMU. The code will be published at `vision.in.tum.de/dm-vio`

4.1 Introduction

Visual-(inertial) odometry is an increasingly relevant task with applications in robotics, autonomous driving, and augmented reality. A combination of cameras and inertial measurement units (IMUs) for this task is a popular and sensible choice, as they are complementary sensors, resulting in a highly accurate and robust system [39]. In the minimal configuration of a single camera, the IMU can also be used to recover the metric scale. However, the scale is not always observable, the most common degenerate case being movement with a constant velocity [43]. Hence, initialization of such system can take arbitrarily long, depending on the trajectory. Even worse, when initialized prematurely the IMU can in fact worsen the performance. The difficulty of IMU initialization is why stereo-inertial methods have outperformed mono-inertial ones in the past.

Most prior systems [36] [37] [38] initially run visual-only odometry and an IMU initialization in parallel. Once finished, the visual-inertial system is started. This introduces a trade-off for the duration of the initialization period: It should be as short as possible, as no IMU information is used in the main system in the

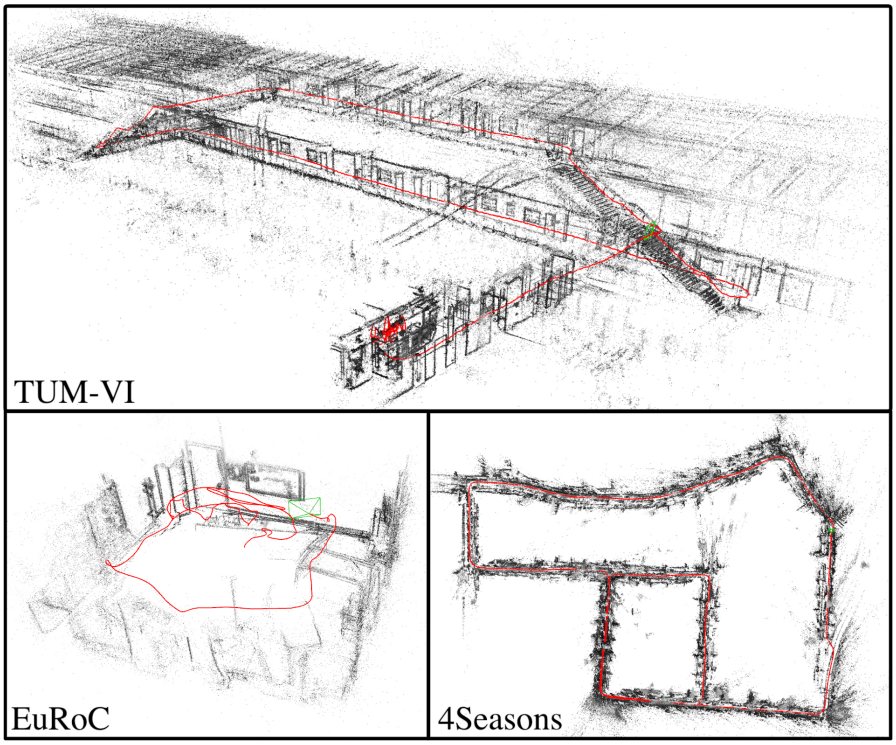


Figure 4.1: In this paper, we propose a novel method for monocular visual-inertial odometry. It provides state-of-the-art performance on three different benchmarks. Here we show pointclouds and trajectories (red) for magistrale5, V203_difficult, and neighbor_2020-03-26_13-32-55_0.

meantime. But when too short, the scale estimate will be inaccurate, leading to bad performance.

VI-DSO [7] instead initializes immediately with an arbitrary scale, and explicitly optimizes the scale in the main system. This yields highly accurate scale estimates, but it can significantly increase the time until the scale is correctly estimated. Also, it can fail in cases where the initial scale error is very high, like in large-scale outdoor environments.

We propose a combination of the two strategies: Similar to the former, we start with a visual-only system and run an IMU initializer in parallel. But after IMU initialization we still estimate scale and gravity direction as explicit optimization variables in the main system. This results in a quickly converging and highly accurate system.

This initialization strategy can lead to three questions: 1) How can the visual uncertainty be properly captured in the IMU initializer. 2) How can information about scale and IMU variables be transferred from the IMU initializer to the main system? 3) If the scale estimate changes, how can a consistent marginalization prior be maintained? VI-DSO [7] tried to address 3 by introducing dynamic

marginalization, which does keep the marginalization factor consistent, but loses too much information in the process.

In this work we propose delayed marginalization, which provides a meaningful answer to all three of these questions. The idea is to maintain a second, *delayed* marginalization prior, which has very little overhead, but enables three core techniques:

1. We can populate the delayed factor graph with new IMU factors to perform the proposed pose graph bundle adjustment (PGBA). This is the basis of an IMU initialization which captures the full photometric uncertainty, leading to increased accuracy.

2. The graph used for IMU initialization can be *re-advanced*, providing a marginalization prior with IMU information for the main system.

3. When the scale changes significantly in the main system we can trigger *marginalization replacement*.

The combination of these techniques makes for a highly accurate initializer, which is robust even to long periods of unobservability. Based on it we implement a visual-inertial odometry (VIO) system featuring a photometric front-end integrated with a new dynamic photometric weight.

We evaluate our method on three challenging datasets (Fig. 4.1), capturing three domains: The EuRoC dataset [70] recorded by a flying drone, the TUM-VI dataset [20] captured with a handheld device, and the 4Seasons dataset [13] representing the automotive scenario. The latter features long stretches of constant velocity, posing a particular challenge for mono-inertial odometry.

We show that our system exceeds the state of the art in visual-inertial odometry, even outperforming stereo-inertial methods. In summary our contributions are:

- Delayed marginalization compensates drawbacks of marginalization while retaining the advantages.

- Pose graph bundle adjustment (PGBA) combines the efficiency of pose graph optimization with the full uncertainty of bundle adjustment.

- A state-of-the-art visual-inertial odometry system with a novel multi-stage IMU initializer and dynamically weighted photometric factors.

The full source code for our approach will be released.

4.2 Related Work

Initially, most visual odometry and SLAM systems have been feature-based [14], either using filtering [29] or nonlinear optimization [25] [24]. More recently, direct

methods have been proposed, which optimize a photometric error function and can operate on dense [46] [30], semi-dense [32], or sparse point clouds [15].

Mourikis and Roumeliotis [39] have shown that a tight integration of visual and inertial measurements can greatly increase accuracy and robustness of odometry. Afterwards, many tightly-coupled visual-inertial odometry [16] [40] and SLAM systems [34] [35] [36] [38] have been proposed.

Initialization of monocular visual-inertial systems is not trivial, as sufficient motion is necessary for the scale to become observable [42] [43]. Most systems [37] [36] [38] start with a visual-only system and use its output for a separate IMU initialization. In contrast to these systems, we continue optimizing the scale explicitly in the main system. We note that ORB-SLAM3 [38] also continues to refine the scale after initialization, but this is a separate optimization fixing all poses and only performed until 75 seconds after initialization. [71] also continues to optimize the scale in the main system, but in contrast to us they do not transfer covariances between the main system and the initializer, thus they do not achieve the same level of accuracy. Different to all these systems, the proposed delayed marginalization allows our IMU initializer to capture the full visual uncertainty and continuously optimize the scale in the main system.

VI-DSO [7] initializes immediately with an arbitrary scale and explicitly optimizes the scale in the main system. It also introduced dynamic marginalization to handle the consequential large scale changes in the main system. Compared to it we propose a separate IMU initializer, delayed marginalization as a better alternative to dynamic marginalization, a dynamic photometric error weight, and more improvements, resulting in greatly improved accuracy and robustness.

4.3 Method

4.3.1 Notation

We denote vectors as bold lowercase letters $\mathbf{x}$, matrices as bold upper-case letter $\mathbf{H}$, scalars as lowercase letters λ, and functions as uppercase letters E. $\mathbf{T}^V_{\text{w\_cam}_i} \in \mathbf{SE}(3)$ represents the transformation from camera i to world in the visual coordinate frame V, and $\mathbf{R}^V_{\text{w\_cam}_i} \in \mathbf{SO}(3)$ is the respective rotation. Poses are represented either in visual frame $\mathbf{P}^V_i := \mathbf{T}^V_{\text{cam}_i\text{\_w}}$, or in inertial frame $\mathbf{P}^I_i := \mathbf{T}^I_{\text{w\_imu}_i}$. If not mentioned otherwise we use poses in visual frame $\mathbf{P}_i := \mathbf{P}^V_i$. We also use states $\mathbf{s}$, which can contain transformations, rotations, and vectors. For states we define the subtraction operator $\mathbf{s}_i \boxminus \mathbf{s}_j$, which applies $\log(\mathbf{R}_i \mathbf{R}_j^{-1})$ for rotations and other Lie group elements, and a regular subtraction for vector values.

4.3.2 Direct Visual-Inertial Bundle Adjustment

The core of DM-VIO is the visual-inertial bundle adjustment performed for all keyframes. As commonly done, we jointly optimize visual and IMU variables in a combined energy function. For the visual part we choose a direct formulation based on DSO [15], as it is a very accurate and robust system. For integrating IMU data into the bundle adjustment we perform preintegration [72] between keyframes.

We optimize the following energy function using the Levenberg-Marquardt algorithm:

$$E(\mathbf{s}) = W(e_{\text{photo}}) \cdot E_{\text{photo}} + E_{\text{imu}} + E_{\text{prior}} \tag{4.1}$$

E_{prior} contains added priors on the first pose and the gravity direction, as well as the marginalization priors explained in section 4.3.3. In the following we describe the individual energy terms and the optimized state.

Photometric error: The photometric energy is based on [15]. We optimize a set of active keyframes $\mathcal{F}$, each of which hosts a set of points $\mathcal{P}_i$. Every point $\mathbf{p}$ is projected into all keyframes $\text{obs}(\mathbf{p})$ where it is visible, and the photometric energy is computed:

$$E_{\text{photo}} = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j} \tag{4.2}$$

$$E_{\mathbf{p}j} = \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{p}} \omega_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma \tag{4.3}$$

For details regarding the variables we refer the reader to [15].

Dynamic photometric weight: In cases of bad image quality, the system should rely mostly on the inertial data. However due to the photometric cost function used, bad image quality will often lead to very large photometric residuals, effectively increasing the photometric weight compared to the IMU. To counteract this we propose a dynamic photometric weight $W(e_{\text{photo}})$. We compute it using the root mean squared photometric error $e_{\text{photo}} = \sqrt{E_{\text{photo}}/n_{\text{residuals}}}$.

$$W(e_{\text{photo}}) = \lambda \cdot \begin{cases} (\theta / e_{\text{photo}})^2, & \text{if } e_{\text{photo}} \geq \theta \\ 1, & \text{otherwise} \end{cases} \tag{4.4}$$

where λ is a static weight component, and θ is the threshold from which the error-dependent weight is activated. This effectively normalizes the root mean squared photometric error to be $\sqrt{\lambda}\theta$ at maximum, similar to a threshold robust cost function [65]. In contrast to the Huber norm in Equation (4.3), which downweights individual points that violate the photometric assumption, this weight addresses cases where the overall image quality is bad and increases the relative weight of the IMU. In our experiments we choose $\theta = 8$.

Optimized variables: We optimize scale and gravity direction as explicit variables. While bundle adjustment can in principle also change the scale and global orientation, convergence is improved when optimizing them explicitly instead [7]. To facilitate this, we represent poses for the visual factors in visual frame V and poses for the IMU factors in IMU frame I. Whereas the IMU frame has a metric scale and a z-axis aligned with gravity direction, the visual frame can have an arbitrary scale and rotation, which is defined during initialization of the visual system. To model this we optimize the scale s and the rotation $\mathbf{R}_{V\_I}$. As yaw is not observable using an IMU, we fix the last coordinate of $\mathbf{R}_{V\_I}$. We convert between the coordinate frames using:

$$\begin{aligned}\mathbf{P}_i^I := \mathbf{T}^I_{\text{w\_imu}_i} = \Omega(\mathbf{P}_i^V, \mathbf{S}, \mathbf{R}_{V\_I}) = \\ \mathbf{R}_{V\_I}^{-1}\mathbf{S}_{I\_V}(\mathbf{P}_i^V)^{-1}\mathbf{S}_{I\_V}^{-1}\mathbf{T}_{\text{cam\_imu}}\end{aligned} \tag{4.5}$$

where $\mathbf{S}_{I\_V}$ is the $\mathbf{Sim}(3)$ element with identity rotation and translation, and scale s. The other variables are converted to $\mathbf{Sim}(3)$, but note that the result has scale 1 and is in $\mathbf{SE}(3)$.

The full state optimized is

$$\mathbf{s} = \{s, \mathbf{R}_{V\_I}\} \cup \bigcup_{i \in \mathcal{F}} \mathbf{s}_i \tag{4.6}$$

with $\mathbf{s}_i$ being the states for all active keyframes defined as:

$$\mathbf{s}_i = \{\mathbf{P}_i^V, \mathbf{v}_i, \mathbf{b}_i, a_i, b_i, d_i^0, d_i^2, ...d_i^j\} \tag{4.7}$$

where $\mathbf{v}_i$ is the velocity, $\mathbf{b}_i$ the bias, a_i and b_i are affine brightness parameters, and d_i^j are the inverse depths of active points hosted in the keyframe. Optimization is performed with a custom integration of the SIMD-accelerated code from [15] for photometric residuals and GTSAM for other factors.

IMU Error: We apply the well-known IMU preintegration first proposed in [73], implemented as smart factors in [74], and further improved in [72]. For this energy we use the IMU state $\mathbf{s}_i^I := \{\mathbf{P}_i^I, \mathbf{v}_i, \mathbf{b}_i\}$, which contains poses in IMU frame and is computed from the optimized state $\mathbf{s}_i$ using Equation (4.5). Given the previous state $\mathbf{s}_i^I$, the preintegration data provides us with a prediction $\widehat{\mathbf{s}}_j^I$ for the following state $\mathbf{s}_j^I$ as well as a covariance matrix $\widehat{\mathbf{\Sigma}}_j$. The resulting inertial error function penalizes deviations of the current state estimate from the predicted state.

$$E_{\text{imu}}(\mathbf{s}_i^I, \mathbf{s}_j^I) := \left(\widehat{\mathbf{s}}_j^I \boxminus \mathbf{s}_j^I\right)^T \widehat{\mathbf{\Sigma}}_j^{-1} \left(\widehat{\mathbf{s}}_j^I \boxminus \mathbf{s}_j^I\right) \tag{4.8}$$

4.3.3 Partial Marginalization using the Schur Complement

We marginalize old variables using the Schur complement. When marginalizing a set β of variables, we gather all factors dependent on them as well as the connected

variables α, which form the Markov blanket. These factors are linearized at the current state estimate, yielding the linear system:

$$\begin{bmatrix} \mathbf{H}_{\alpha\alpha} & \mathbf{H}_{\alpha\beta} \\ \mathbf{H}_{\beta\alpha} & \mathbf{H}_{\beta\beta} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{\alpha} \\ \mathbf{s}_{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\alpha} \\ \mathbf{b}_{\beta} \end{bmatrix} \tag{4.9}$$

We apply the Schur-complement, which results in the new linear system $\widehat{\mathbf{H}_{\alpha\alpha}}\boldsymbol{s}_{\alpha} = \widehat{\boldsymbol{b}_{\alpha}}$ with

$$\widehat{\mathbf{H}_{\alpha\alpha}} = \mathbf{H}_{\alpha\alpha} - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\mathbf{H}_{\beta\alpha} \tag{4.10}$$

$$\widehat{\boldsymbol{b}_{\alpha}} = \boldsymbol{b}_{\alpha} - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\boldsymbol{b}_{\beta} \tag{4.11}$$

This linear system forms a marginalization prior connecting all variables in α (Fig. 4.2a).

We keep a maximum of $N_f = 8$ keyframes during the bundle adjustment[1]. The marginalization strategy is taken over from [15]: This means that different from a fixed-lag smoother, we do not always marginalize the oldest pose, but instead keep a combination of newer and older poses, as long as they do not leave the field of view. As shown in [15] this is superior to a fixed-lag smoother for visual odometry. When marginalizing a pose, first all remaining points hosted in the frame are marginalized and residuals with remaining active points are dropped. This retains sparsity of the Hessian while preserving enough information.

4.3.4 Delayed Marginalization

The concept of marginalization explained in the previous section has the advantage of capturing the full probability distribution. In fact, solving the resulting smaller system is equivalent to solving the much larger original system, as long as the marginalized factors are not relinearized.

However, it also comes with severe drawbacks: Reverting the marginalization of a set of variables is not possible without redoing the whole marginalization procedure. Also, to keep the marginalization prior consistent, First-Estimates Jacobians (FEJ) [69] have to be applied. This means that the linearization point of all connected variables has to be fixed as soon as they are connected to a marginalization prior. This is especially problematic for visual-inertial odometry, where the scale is connected to the marginalization prior as soon as the first keyframe is marginalized, but might change significantly. In [7] dynamic marginalization was introduced to combat this, but it is limited in its application to a single one-dimensional variable, namely the scale, and loses most prior inertial information when the scale changes quickly.

Here we introduce delayed marginalization which circumvents the drawbacks of marginalization while retaining the advantages. It enables us to:

[1] We define N_f as the maximum number of frames during bundle adjustment, whereas in [15] it is the number of frames after marginalization.

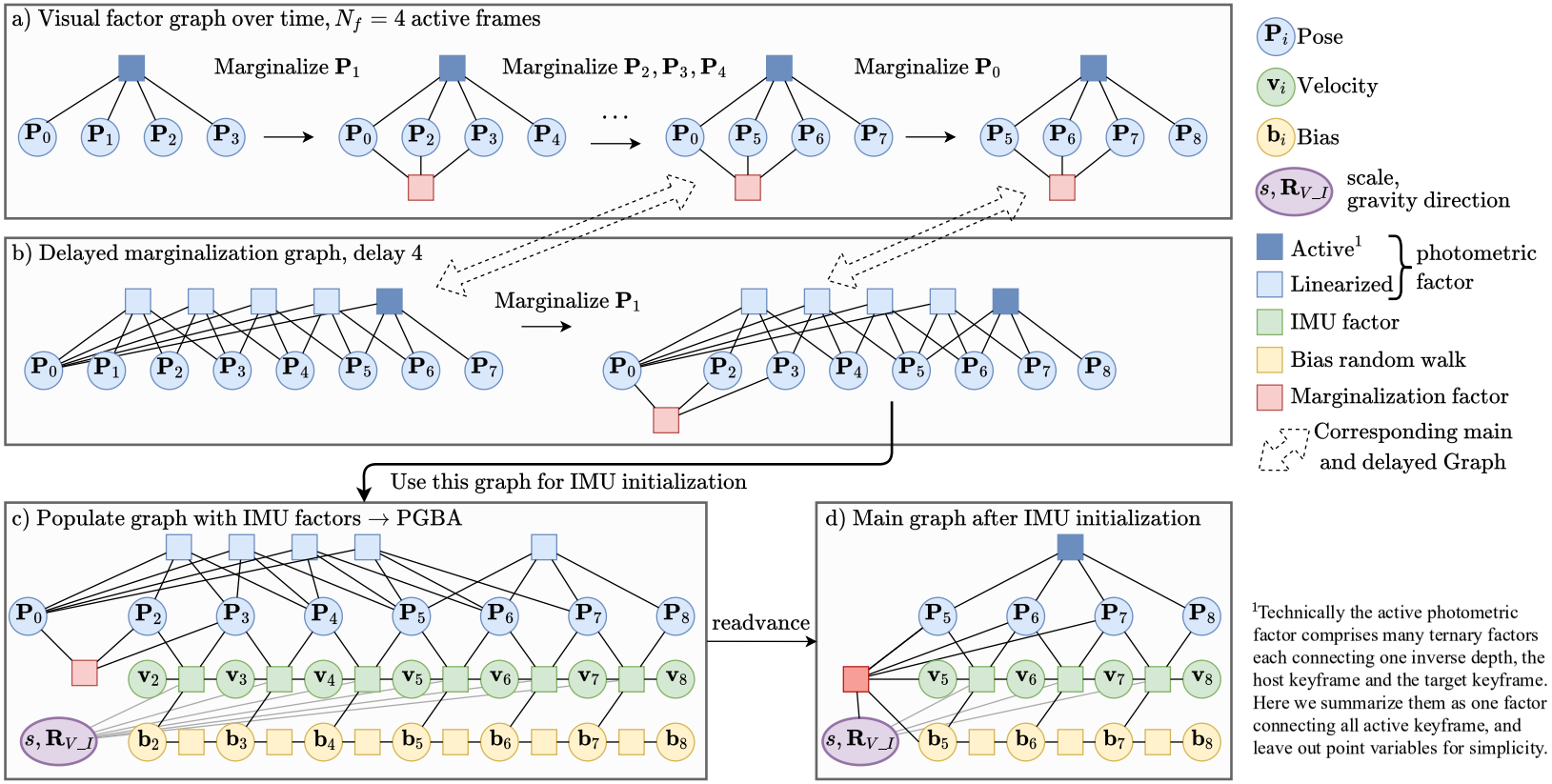


Figure 4.2: Delayed Marginalization and PGBA: a) Normal marginalization in the visual graph. Note that not always the oldest pose is marginalized. b) Delayed marginalization: We marginalize all variables in the same order as the main graph, but with a delay d (in practice $d = 100$). Marginalization in this graph is equally fast as marginalizing in the main graph. c) For the pose graph bundle adjustment (PGBA) we populate the delayed graph with IMU factors. This optimization leverages the full photometric uncertainty. d) We readvance the marginalization in the graph used for PGBA to obtain an updated marginalization prior for the main system. This transfers inertial information from the initializer to the main system.

- Effectively undo part of the marginalization to capture the full photometric probability distribution for the pose graph bundle adjustment (section 4.3.5).
- Update the initially visual-only marginalization prior with IMU information after the IMU initialization.
- Relinearize variables in the Markov blanket while keeping all visual and most inertial information.

The idea of delayed marginalization is that marginalization cannot be undone, but it can be delayed: *In addition* to the normal marginalization prior we also maintain a second, *delayed* marginalization prior and corresponding factor graph. In this delayed graph, marginalization of frames is performed with a delay of d. Points are still marginalized at the same time in the delayed graph, resulting in linearized photometric factors. We note that the same marginalization order as in the original graph is preserved. Switching to a fixed-lag smoother for this graph would immediately lead to a much larger Markov blanket jeopardizing the runtime of the system. E.g. in Fig. 4.2b we depict the delayed marginalization of $\mathbf{P}_1$. The Markov blanket only contains $\mathbf{P}_0$, $\mathbf{P}_2$, and $\mathbf{P}_3$. If we instead marginalized the oldest frame $\mathbf{P}_0$, the Markov blanket would contain $\mathbf{P}_1 - \mathbf{P}_7$, leading to higher runtime.
Marginalization in the delayed graph has the same runtime as marginalization in the original graph. The delayed graph contains the same photometric factors as the original graph, and points are marginalized at the same time. This means that each linearized photometric factor in the delayed graph is connected to exactly the $N_f = 8$ keyframes which were active when the respective factor was generated. By keeping the marginalization order, the Markov blanket in the delayed graph always has the same size as the one in the original graph. Thus, the runtime of the Schur complement is the same. This means that the overhead of Delayed Marginalization is very small even for arbitrarily large delays, as it only amounts to an additional marginalization procedure per delayed graph.

4.3.5 Pose Graph Bundle Adjustment for IMU Initialization

PGBA utilizes delayed marginalization for IMU initialization. The idea is to populate the delayed graph with IMU factors and optimize all variables (Fig. 4.2c).
Populating the graph: Let a frame $\mathbf{P}_i$ be directly connected to the newest pose $\mathbf{P}_k$ iff all poses $\mathbf{P}_j, i < j < k$ have not been marginalized yet. We determine the first frame $\mathbf{P}_{\text{conn}}$ in the delayed graph which is still directly connected to the newest frame. In Fig. 4.2c this is $\mathbf{P}_2$. From there, we insert IMU factors and bias factors to all successive frames.

We cannot start before $\mathbf{P}_{\text{conn}}$ because we do not want to insert IMU factors between non-successive keyframes. As the marginalization order is not fixed-lag, this means that we have to optimize poses without corresponding IMU variables.

It can be shown that there can be at most $N_f - 2$ poses without IMU variables. The reason is that all non-connected poses were at some point active at the same time. This means that in practice we have at least $d - N_f + 2$ poses for which we can add IMU data. In practice, we choose $N_f = 8$ and delay $d = 100$, meaning that even in the worst case there will be 93 IMU factors in the optimization. As explained previously, fixed-lag smoothing would either result in a dense Hessian or in suboptimal performance of the visual system, so this is a very good trade-off.

Optimization: We optimize the graph with the GTSAM [68] library using the Levenberg-Marquardt optimizer with the provided Ceres-default settings. In this optimization all points are marginalized. We call it pose graph bundle adjustment because it is a combination of regular pose graph optimization (PGO) and bundle adjustment (BA). In contrast to BA, we do not update our estimates for point depths and do not relinearize photometric error terms. Different from PGO we do not use binary constraints between poses, but instead use "octonary" constraints, which connect N_f frames and capture the full probability distribution of BA. Compared to PGO our solution is thus more accurate while being much faster than full BA. By using a fixed delay it is also constrained in runtime even though it can be performed at any time without losing any prior visual information.

Readvancing: Another advantage of delayed marginalization and our PGBA is that we can obtain a marginalization prior for the main system, capturing all the visual and inertial information. For this, we readvance the graph used for the PGBA. This works by successively marginalizing all the variables which have been marginalized in the main graph. Again, this is done preserving the marginalization order, which means that in each marginalization step the Markov blanket has a fixed maximum size. Hence, marginalizing step by step is significantly faster than marginalizing all variables at once, which would involve a much larger matrix inversion. Fig. 4.2d shows the result of readvancing.

4.3.6 Robust Multi-Stage IMU Initialization

Our initialization strategy is based on three insights:

1. When some variables are unknown (in our case scale, gravity direction, and biases) and others are close to the optimum, it is most efficient to first optimize only the unknown variables and fix the others.

2. The most accurate result can be obtained by optimizing all variables jointly, capturing the full covariance.

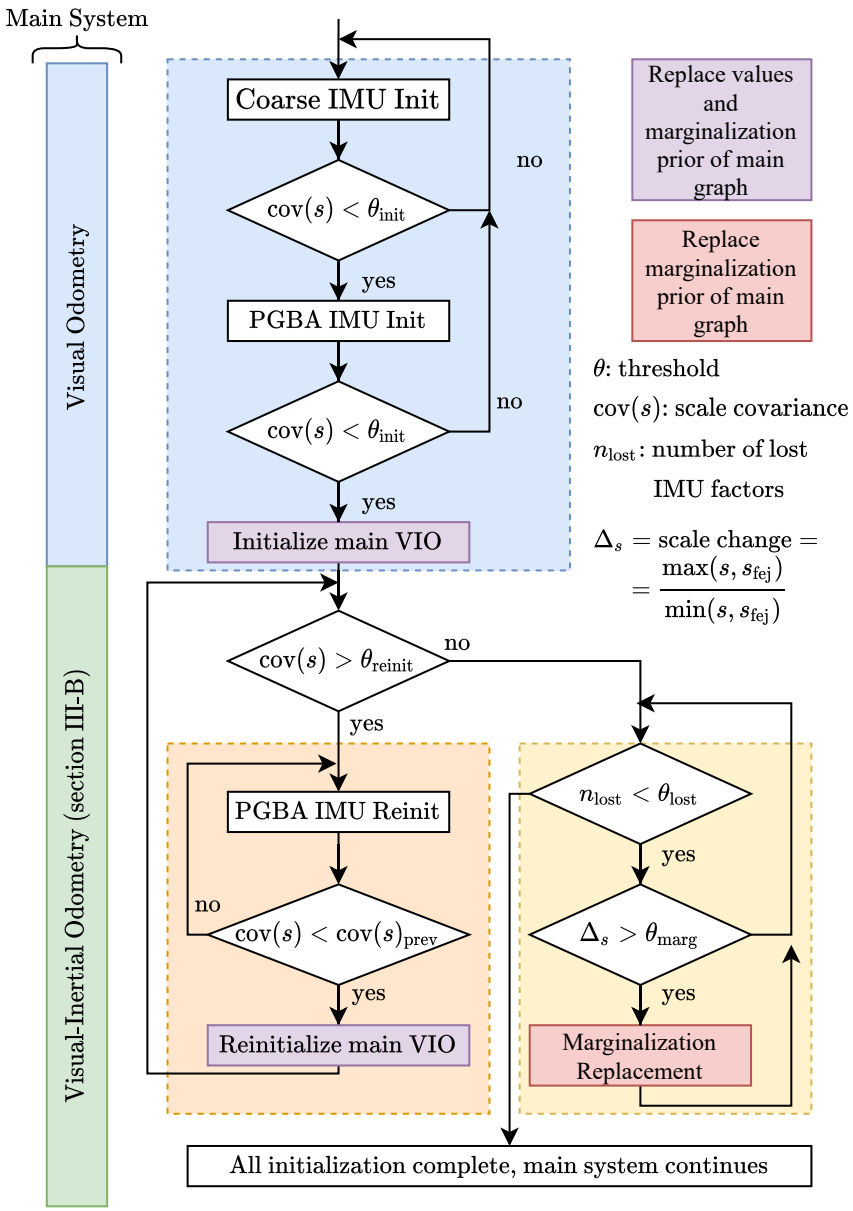


Figure 4.3: Our multi-stage IMU initialization. First we perform a coarse IMU initialization, which provides initial values for the PGBA. The PGBA captures the full visual covariances, achieving very accurate initial estimates for scale, gravity direction, and biases. It also provides an updated marginalization prior for the main graph. By also optimizing the scale in the main VIO system (green box), we can initialize early (purple box) and later reinitialize or perform marginalization replacement, if new information about the scale becomes available. The proposed delayed marginalization is what enables both, the PGBA, and the marginalization replacement.

3. When marginalizing, connected variables have to be close to the optimum, otherwise the marginalization prior becomes inconsistent.

These observations inspire 1) the Coarse IMU Initialization, 2) the PGBA, and 3) the Marginalization Replacement (Fig. 4.3). Note that after "Initialize main VIO", the main VIO system 4.3.2 (green box) is already running in parallel.

For this initializer we use a single delayed graph with a delay of $d = 100$. This delayed graph will always contain only visual factors and no IMU factors, even after the first initialization, to facilitate the marginalization replacement.

Coarse IMU Initialization: For this we only consider the last $d = 100$ keyframes and connect them with IMU factors. Similar to the inertial only optimization used for initialization in ORB-SLAM3 [38], in this optimization we fix the poses and use a single bias. We only optimize velocities, bias, the gravity direction and the scale. Gravity direction is initialized by averaging the accelerometer measurements between the first two keyframes, scale is initialized with 1, and bias and velocity with 0. This optimization is less accurate than PGBA but serves as an initialization for it. After optimizing, we compute the marginal covariance for the scale $\text{cov}(s)$ and continue to the PGBA if it is smaller than a threshold θ_{init}. As shown in [75], taking into account IMU noise parameters is crucial for good IMU initialization, which our coarse IMU initialization satisfies. But for our method it is just an initialization for the PGBA, which in addition models photometric noise properties.

PGBA IMU Init.: We perform PGBA as explained in section 4.3.5. Afterwards, we again threshold on the marginal covariance for the scale to find out if the optimization was successful. When a tighter threshold θ_{reinit} is not also met, we initialize with the result, but will perform another PGBA afterwards to reinitialize with more accurate values. This reinitialization enables us to set θ_{init} to a relatively large value, allowing to use IMU data in the main system earlier.

Marginalization Replacement: After IMU initialization, we monitor how much the scale s changes compared to the First-Estimates scale s_{fej} used in the marginalization prior. If this change exceeds a threshold θ_{marg}, i.e. $\delta_s := \max(s, s_{\text{fej}})/\min(s, s_{\text{fej}}) > \theta_s$, we trigger a marginalization replacement. For the marginalization replacement we rebuild the PGBA graph by populating the delayed graph with IMU factors, Fig. 4.2c). Different from the PGBA, we do not optimize in this graph but instead just readvance it to obtain an updated marginalization prior. This new prior still contains all visual factors and at least the last $d - N_f + 1 = 93$ IMU factors. We disable the marginalization replacement if more than $\theta_{\text{lost}} = 50\%$ of the IMU factors contained in the previous prior would be lost. This procedure shows how delayed marginalization can be used to update FEJ values, overcoming one of the main problems of marginalization.

In realtime mode we perform the coarse IMU initialization and the PGBA in a separate thread. Note how important the proposed delayed marginalization is for this IMU initialization. It allows the PGBA to capture the full covariance from the

photometric bundle adjustment. By readvancing, this also enables us to generate a marginalization prior for the main system, containing all IMU information from the initializer. Lastly, it is used for updating the marginalization prior when the scale changes after the initialization.

4.4 Results

We evaluate our method on the EuRoC dataset [70], the TUM-VI dataset [20], and the 4Seasons dataset [13], covering flying drones, handheld sequences, and autonomous driving respectively. We encourage the reader to watch the supplementary video which shows qualitative realtime results on 4Seasons and TUM-VI slides1. We also provide ablation studies and runtime evaluations in the supplementary available at `vision.in.tum.de/dm-vio`.

Unless otherwise stated all experiments are performed in realtime mode on the same MacBook Pro 2013 (i7 at 2.3GHz) which was used for generating the results in [7], without utilizing the GPU. As ORB-SLAM3 is not officially supported on MacOS, we show results for it on a slightly stronger desktop with an Intel Core i7-7700K at 4.2GHz, which is very similar to the PC used in their paper.

All methods are evaluated 10 times for EuRoC and 5 times for the other datasets on each sequence. Following [15], results are presented in cumulative error plots, which show how many sequences (y-axis) have been tracked with an accuracy better than the threshold on the x-axis. We perform $\mathbf{SE}(3)$ alignment of the trajectory with the provided ground-truth and report the root mean squared error (RMSE), also called absolute trajectory error (ATE). On TUM-VI and 4Seasons, trajectory lengths can vary greatly so we report the drift in %, which we compute with $\text{drift} = \frac{\text{rmse} \cdot 100}{\text{length}}$. We also show tables to compare to numbers from other papers and report the median result for each sequence for our method.

4.4.1 EuRoC dataset

The EuRoC dataset [70] is the most popular visual-inertial dataset to date, and many powerful methods have been evaluated on it. In Table 4.1 we compare to the state-of-the art in visual-inertial odometry, all results are without loop-closure. Our method outperforms all other methods clearly in terms of RMSE. The closest competitor is Basalt [34], a stereo-inertial method which achieves a smaller error on 2 sequences. We also observe the lowest average scale error reported on the dataset so far, confirming that our contributions in IMU initialization have a positive impact on performance. In the supplementary we provide runtime evaluations, showing that tracking takes 10.34ms on average, and keyframe processing takes 53.67ms. The delayed marginalization is responsible for an overhead of 0.44ms or 0.8% in the keyframe thread.

Table 4.1: Evaluation of various mono (M) and stereo (S) visual-inertial odometry systems on EuRoC. Our system provides a notable improvement over the state-of-the art. Please note that a full SLAM system utilizing loop closures can achieve even more accurate results, e.g. ORB-SLAM-VI has a mean error of 0.075, and ORB-SLAM3 has a mean error of 0.043.

Sequence		MH1	MH2	MH3	MH4	MH5	V11	V12	V13	V21	V22	V23	Avg
MCSKF[2] [39] (M)	RMSE	0.42	0.45	0.23	0.37	0.48	0.34	0.20	0.67	0.10	0.16	1.13	0.414
OKVIS[1] [40] (M)	RMSE	0.33	0.37	0.25	0.27	0.39	0.094	0.14	0.21	0.090	0.17	0.23	0.231
ROVIO[2] [16] (M)	RMSE	0.21	0.25	0.25	0.49	0.52	0.10	0.10	0.14	0.12	0.14	0.14	0.224
VINS-Mono [36] (M)	RMSE	0.15	0.15	0.22	0.32	0.30	0.079	0.11	0.18	0.080	0.16	0.27	0.184
Kimera [35] (S)	RMSE	0.11	0.10	0.16	0.24	0.35	0.05	0.08	0.07	0.08	0.10	0.21	0.141
Online VIO [71] (M)	RMSE	0.14	0.13	0.20	0.22	0.20	0.05	0.07	0.16	0.04	0.11	0.17	0.135
VI-DSO [7] (M)	RMSE	**0.062**	**0.044**	0.117	0.132	0.121	0.059	0.067	0.096	0.040	0.062	0.174	0.089
	Scale Error (%)	1.1	0.5	0.4	0.2	0.8	1.1	1.1	0.8	1.2	0.3	0.4	0.7
BASALT [34] (S)	RMSE	0.07	0.06	**0.07**	0.13	0.11	**0.04**	0.05	0.10	0.04	**0.05**	-	0.072
DM-VIO (M)	RMSE	0.065	**0.044**	0.097	**0.102**	**0.096**	0.048	**0.045**	**0.069**	**0.029**	**0.050**	**0.114**	**0.069**
	Scale Error (%)	1.3	0.9	0.4	0.2	0.4	0.4	1.0	0.3	0.02	0.6	0.8	0.6

[1] results taken from [36].
[2] results taken from [76], these are **Sim**(3)-aligned.
All other results are taken from the respective paper.

Table 4.2: RMSE ATE in m on the TUM-VI dataset [20]. Best results in bold, underline is the best result among monocular methods. DM-VIO outperforms even state-of-the-art stereo-inertial methods by a large margin.

Sequence	ROVIO stereo	VINS mono	OKVIS stereo	BASALT stereo	DM-VIO mono	length [m]
corridor1	0.47	0.63	0.33	0.34	**0.19**	305
corridor2	0.75	0.95	0.47	**0.42**	0.47	322
corridor3	0.85	1.56	0.57	0.35	**0.24**	300
corridor4	**0.13**	0.25	0.26	0.21	**0.13**	114
corridor5	2.09	0.77	0.39	0.37	**0.16**	270
magistrale1	4.52	2.19	3.49	**1.20**	2.35	918
magistrale2	13.43	3.11	2.73	**1.11**	2.24	561
magistrale3	14.80	**0.40**	1.22	0.74	1.69	566
magistrale4	39.73	5.12	**0.77**	1.58	1.02	688
magistrale5	3.47	0.85	1.62	**0.60**	0.73	458
magistrale6	X	2.29	3.91	3.23	**1.19**	771
outdoors1	101.95	**74.96**	X	255.04	123.24	2656
outdoors2	21.67	133.46	73.86	64.61	**12.76**	1601
outdoors3	26.10	36.99	32.38	38.26	**8.92**	1531
outdoors4	X	16.46	19.51	17.53	**15.25**	928
outdoors5	54.32	130.63	13.12	7.89	**7.16**	1168
outdoors6	149.14	133.60	96.51	65.50	**34.86**	2045
outdoors7	49.01	21.90	13.61	**4.07**	5.00	1748
outdoors8	36.03	83.36	16.31	13.53	**2.11**	986
room1	0.16	0.07	0.06	0.09	**0.03**	146
room2	0.33	**0.07**	0.11	**0.07**	0.13	142
room3	0.15	0.11	**0.07**	0.13	0.09	135
room4	0.09	0.04	**0.03**	0.05	0.04	68
room5	0.12	0.20	0.07	0.13	**0.06**	131
room6	0.05	0.08	0.04	**0.02**	**0.02**	67
slides1	13.73	0.68	0.86	0.32	**0.31**	289
slides2	0.81	0.84	2.15	**0.32**	0.87	299
slides3	4.68	0.69	2.58	0.89	**0.60**	383
avg drift%	16.83*	1.700	0.815*	0.939	**0.472**	normalized

4.4.2 TUM-VI dataset

The TUM-VI dataset [20] is a very challenging handheld dataset, featuring large-scale indoor and outdoor scenes, and even sequences sliding down a tube, where almost the full image is covered. With long periods of walking in straight lines, stereo methods have an advantage here as they still can observe the scale with constant motion. We compare to the state-of-the-art visual-inertial odometry methods evaluated in [20] in Table 4.2. Our method clearly outperforms the other monocular method in VINS-Mono [36] on most sequences, and even compared to the stereo methods it shows the best result on 16 sequences and a mean drift of 0.472. The closest competitor is again Basalt, which achieves the best result on 8 sequences and a mean drift of 0.939.

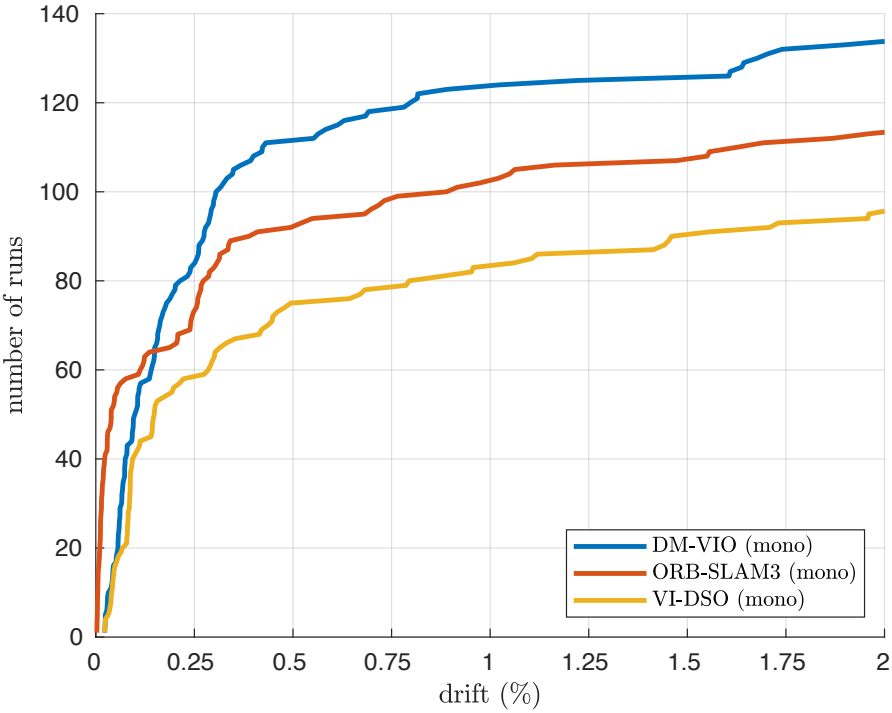


Figure 4.4: Cumulative error plot for the TUM-VI dataset (drift in %). Our method clearly outperforms both VI-DSO and ORB-SLAM3 in terms of robustness. Thanks to its powerful loop closure system, ORB-SLAM3 has an advantage in terms of accuracy on some sequences.

On this dataset, we also evaluate against ORB-SLAM3 [38], which is the state-of-the art visual-inertial SLAM system. This is not entirely fair as ORB-SLAM3 uses loop closures (which cannot be disabled), constituting an advantage over the other methods. We find the comparison still helpful as it allows to make conclusions regarding the underlying odometry. We have evaluated ORB-SLAM3 5 times on each sequence and reproduced their results with code and settings provided by the authors. For this comparison we have also evaluated VI-DSO [7], and the results are shown in Fig. 4.4. We observe that ORB-SLAM3 is more accurate on some sequences thanks to its very strong loop closure system. However, our method is more robust overall. This indicates that an integration of loop closure and map reuse into our system would be an interesting future research direction.

4.4.3 4Seasons dataset

The 4Seasons dataset [13] is a very recent automotive dataset, which, in contrast to most other car datasets, features a well time-synchronized visual-inertial sensor. The lower part of the images is obstructed by the car hood, hence we crop off

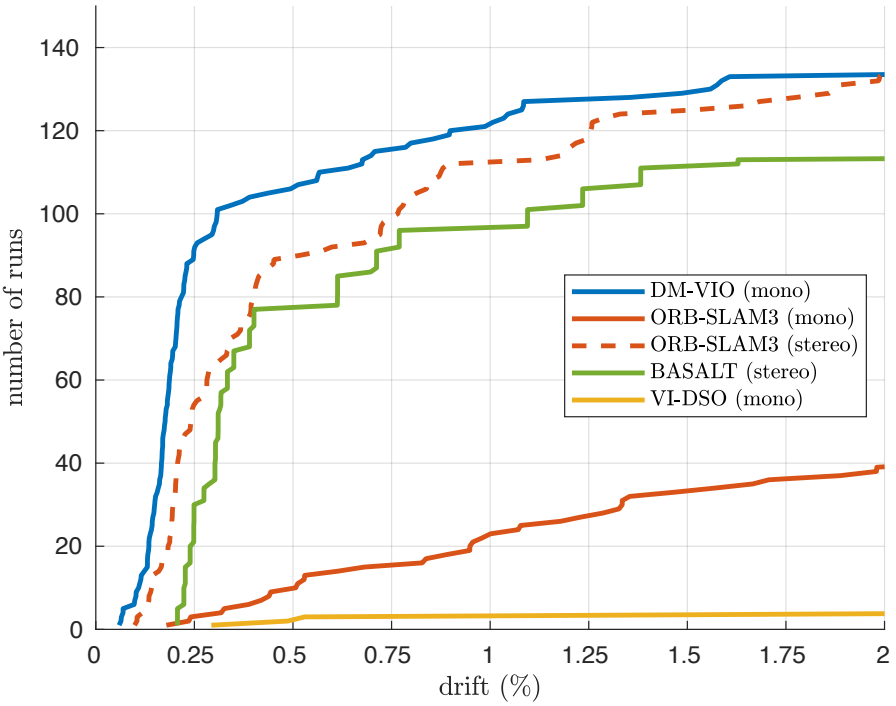


Figure 4.5: Cumulative error plot for the 4Seasons dataset (drift in %). With lots of stretches with constant velocity, this dataset is extremely challenging for monocular visual-inertial methods. Thanks to our novel IMU initializer powered by delayed marginalization and PGBA, DM-VIO is able to cope with it and even outperforms stereo-inertial methods.

the bottom 96 pixels, which we do for all methods. As this is the first odometry method to evaluate on the 4Seasons dataset, we make sure to determine IMU noise parameters for all methods the same way to ensure a fair comparison: We have manually read off the accelerometer and gyroscope noise density and bias random walk from the Allan variance plot provided in the data sheet of the IMU. To handle unmodeled effects we follow [20] and inflate noise values by different amounts to determine the best setting for all methods. For each method we tried noise models inflated by 1, 10, 100, 1000 respectively and chose the configuration which gave best results. For VI-DSO and for our method we slightly modified the visual initializer by adding a zero-prior to the translation on the x and y axis, and also added a threshold to stop keyframe creation for translations smaller than 0.01m (the latter was not activated for VI-DSO as it did not improve the results for it). Otherwise, parameters are the same as for the other experiments. For Basalt we tried all three provided default configurations with the optimal noise values to find the best settings. After choosing the configuration for each method, we perform one final evaluation, running all 30 sequences 5 times each.

The results are shown in Fig. 4.5. It is clear that the automotive scenario is

very challenging for monocular methods. This is expected as it naturally features many stretches with constant motion, where scale is not observable, constituting a challenge for IMU initialization. Thanks to our novel IMU initialization, DM-VIO not only works well on the dataset but even outperforms stereo-inertial ORB-SLAM3 and Basalt, while using monocular images and no loop closures.

4.5 Conclusion and Future Work

We have presented a monocular visual-inertial odometry system which outperforms the state of the art, even stereo-inertial methods. Thanks to a novel IMU initializer, it works well in flying, handheld, and automotive scenarios, extending the applicability of monocular methods. The foundation of our IMU initialization is delayed marginalization, which also enables the pose graph bundle adjustment.

We anticipate that this method will spark further research in this direction. The idea of delayed marginalization could be applied to more use cases, e.g. for reactivating old keyframes in a marginalization setting to enable map reuse. The pose graph bundle adjustment can also be applied to long-term loop closures. Lastly, our open-source system is easily extendible, as all optimizations are integrated with GTSAM, allowing to quickly add new factors. This could be used for GPS integration, wheel odometry, and more.

Chapter 5

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

Authors	Lukas von Stumberg[1,2*]	stumberg@in.tum.de
	Patrick Wenzel[1,2*]	wenzel@in.tum.de
	Qadeer Khan[1,2]	qadeer.khan@tum.de
	Daniel Cremers[1,2]	cremers@in.tum.de

[*]These authors contributed equally

[1]Technical University of Munich, Munich, Germany
[2]Artisense

Publication

L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers, **GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization**, *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 5, no. 2, pp. 890–897, 2020. DOI: 10.1109/LRA.2020.2965031. arXiv: 1904.11932

Contribution		
	Problem definition	*significantly contributed*
	Literature survey	*significantly contributed*
	Algorithm development	*significantly contributed*
	Method implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

Abstract Direct SLAM methods have shown exceptional performance on odometry tasks. However, they are susceptible to dynamic lighting and weather changes while also suffering from a bad initialization on large baselines. To overcome this, we propose GN-Net: a network optimized with the novel Gauss-Newton loss for training weather invariant deep features, tailored for direct image alignment. Our network can be trained with pixel correspondences between images taken from different sequences. Experiments on both simulated and real-world datasets demonstrate that our approach is more robust against bad initialization, variations in daytime, and weather changes thereby outperforming state-of-the-art direct and indirect methods. Furthermore, we release an evaluation benchmark for relocalization tracking against different types of weather. Our benchmark is available at `https://vision.in.tum.de/gn-net`.

5.1 Introduction

In recent years, very powerful visual SLAM algorithms have been proposed [24], [25]. In particular, direct visual SLAM methods have shown great performance, outperforming indirect methods on most benchmarks [15], [32], [47]. They directly leverage the brightness data of the sensor to estimate localization and 3D maps rather than extracting a heuristically selected sparse subset of feature points. As a result, they exhibit a boost in precision and robustness. Nevertheless, compared to indirect methods, direct methods suffer from two major drawbacks:

1. Direct methods need a good initialization, making them less robust for large baseline tracking or cameras with a low frame rate.

2. Direct methods cannot handle changing lighting/weather conditions. In such situations, their advantage of being able to pick up very subtle brightness variations becomes a disadvantage to the more lighting invariant features.

In the last years, researchers have tackled the multiple-daytime tracking challenge with deep learning approaches that are designed to convert nighttime images to daytime images *e.g.* using GANs [77]–[79]. While this improves the robustness to changing lighting, one may ask why images should be the best input representation. Could there be better alternate representations?

This paper addresses the problem of adapting direct SLAM methods to challenging lighting and weather conditions. In this work, we show how to convert images into a multi-dimensional feature map which is invariant to lighting/weather changes and has by construction a larger basin of convergence. Thereby we overcome the aforementioned problems simultaneously. The deep features are trained with a novel Gauss-Newton loss formulation in a self-supervised manner. We employ a

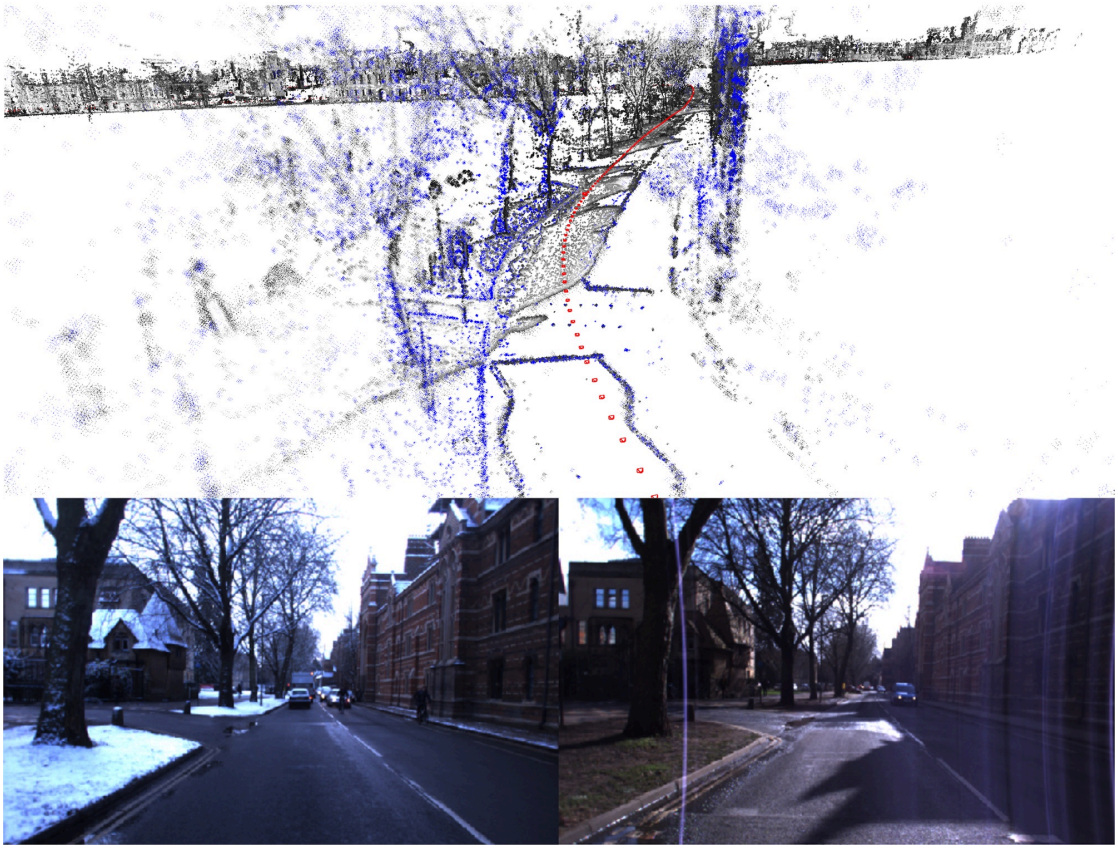


Figure 5.1: We relocalize a snowy sequence from the Oxford RobotCar dataset in a pre-built map created using a sunny weather condition. The points from the prior map (gray) well align with the new points from the current run (blue), indicating that the relocalization is indeed accurate.

Siamese network trained with labels obtained either from simulation data or any state-of-the-art SLAM algorithm. This eliminates the additional cost of human labeling that is typically necessary for training a neural network. We exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm for direct image alignment. For this, we propose the Gauss-Newton loss which is designed to maximize the probability of identifying the correct pixel correspondence. The proposed loss function thereby enforces a feature representation that is designed to admit a large basin of convergence for the subsequent Gauss-Newton optimization. The superiority of our method stems from its ability to generate these multi-channel, weather-invariant deep features that facilitate relocalization across different weathers. Figure 5.1 shows how our method can successfully relocalize a snowy sequence in a pre-built map created using a sunny sequence.

In common benchmarks [28], localizing accurately in a pre-built map has been

tackled by finding nearby images (*e.g.* by using NetVLAD [80]) and tracking the relative pose (6DOF) between them. However, we propose to split this into two separate tasks. In this work, we focus on the second challenge which we refer to as *relocalization tracking*. This way, we can evaluate its performance in isolation. This is formalized to what we refer to as *relocalization tracking*. Since there is no publicly available dataset to evaluate *relocalization tracking* performance across multiple types of weathers, we are releasing an evaluation benchmark having the following 3 attributes:

- It contains sequences from multiple different kinds of weathers.
- Pixel-wise correspondences between sequences are provided for both simulated and real-world datasets.
- It decouples relocalization tracking from the image retrieval task.

The challenge here in comparison with normal pose estimation datasets [19], [70] is that the images involved are usually captured at different daytimes/seasons and there is no good initialization of the pose. We summarize the main contributions of our paper as:

- We derive the Gauss-Newton loss formulation based on the properties of direct image alignment and demonstrate that it improves the robustness to large baselines and illumination/weather changes.
- Our experimental evaluation shows, that GN-Net outperforms both state-of-the-art direct and indirect SLAM methods on the task of *relocalization tracking*.
- We release a new evaluation benchmark for the task of *relocalization tracking* with ground-truth poses. It is collected under dynamic conditions such as illumination changes, and different weathers. Sequences are taken from the the CARLA [81] simulator as well as from the Oxford RobotCar dataset [82].

5.2 Related Work

We review the following main areas of related work: visual SLAM, visual descriptor learning, deep direct image alignment, and image-based relocalization in SLAM.
Direct versus indirect SLAM methods: Most existing SLAM systems that have used feature descriptors are based on traditional manual feature engineering, such as ORB-SLAM [24], MonoSLAM [29], and PTAM [25].

An alternative to feature-based methods is to skip the pre-processing step of the raw sensor measurements and rather use the pixel intensities directly. Popular

direct visual methods are DTAM [30], LSD-SLAM [32], DSO [15], and PhotoBundle [47]. However, the main limitation of direct methods is the *brightness constancy* assumption which is rarely fulfilled in any real-world robotic application [83]. The authors of [84] propose to use binary feature descriptors for direct tracking called Bit-planes. While improving the robustness to bad lighting situations it was also found that Bit-planes have a smaller convergence basin than intensities. This makes their method less robust to bad initialization. In contrast, the features we propose *both* improve robustness to lighting and the convergence basin.

Visual descriptor learning: Feature descriptors play an important role in a variety of computer vision tasks. For example, [85] proposed a novel correspondence contrastive loss which allows for faster training and demonstrates their effectiveness for both geometric and semantic matching across intra-class shape or appearance variations. In [86], a deep neural network is trained using a contrastive loss to produce viewpoint- and lighting-invariant descriptors for single-frame localization. The authors of [87] proposed a CNN-based model that learns local patterns for image matching without a global geometric model. [88] uses convolutional neural networks to compute descriptors which allow for efficient detection of poorly textured objects and estimation of their 3D pose. In [89], the authors propose to train features for optical flow estimation using a Hinge loss based on correspondences. In contrast to our work, their loss function does not have a probabilistic derivation and they do not apply their features to pose estimation. [56] uses deep learning to improve SLAM performance in challenging situations. They synthetically create images and choose the one with most gradient information as the ground-truth for training. In contrast to them, we do not limit our network to output images similar to the real world. In [54], the authors compare dense descriptors from a standard CNN, SIFT, and normal image intensities for dense Lucas-Kanade tracking. There, it can be seen that grayscale values have a better convergence basin than the other features, which is something we overcome with our approach.

Deep direct image alignment: BA-NET [90] introduces a network architecture to solve the structure from motion (SfM) problem via feature-metric bundle adjustment. Unlike the BA-NET, instead of predicting the depth and the camera motion simultaneously, we propose to only train on correspondences obtained from a direct SLAM system. The advantage is that correspondences are oftentimes easier to obtain than accurate ground-truth poses. Furthermore, we combine our method with a state-of-the-art direct SLAM system and utilize its depth estimation, whereas BA-NET purely relies on deep learning. RegNet [91] is another line of work which tries to replace the handcrafted numerical Jacobian by a learned Jacobian with the help of a depth prediction neural network. However, predicting a dense depth map is often inaccurate and computationally demanding. The authors of [92] propose to use a learning-based inverse compositional algorithm for dense image alignment. The drawback of this approach is that the algorithm is very sensitive to the data distribution and constrained towards selecting the right hyperparameters.

In [93] they use high-dimensional features in a direct image alignment framework for monocular VO. In contrast to us, they only use already existing features and do not apply them for relocalization.

Relocalization: An important task of relocalization is to approximate the pose of an image by simply querying the most similar image from a database [94], [95]. However, this has only limited accuracy unless the 6DOF pose between the queried and the current image is estimated in a second step. Typically, this works by matching 2D-3D correspondences between an image and a point cloud and estimating the pose using indirect image alignment [31]. In contrast, we propose to use direct image alignment paired with deep features.

Relocalization benchmarks: The authors of [28] have done sequence alignment on the Oxford RobotCar dataset, however, they have not made the matching correspondences public. The Photo Tourism [96] is another dataset providing images and ground-truth correspondences of popular monuments from different camera angles and across different weather/lighting conditions. However, since the images are not recorded as a sequence, relocalization tracking is not possible. Furthermore, their benchmark only supports the submission of features rather than poses, thereby restricting evaluation to only indirect methods.

5.3 Deep Direct SLAM

In this work, we argue that a network trained to output features which produce better inputs for direct SLAM as opposed to normal images should have the following properties:

- Pixels corresponding to the same 3D point should have similar features.
- Pixels corresponding to different 3D points should have dissimilar features.
- When starting in a vicinity around the correct pixel, the Gauss-Newton algorithm should move towards the correct solution.

For optimizing the last property, we propose the novel Gauss-Newton loss which makes use of the probabilistic background of the Gauss-Newton algorithm for direct image alignment. The final loss is a weighted sum of the pixel-wise contrastive loss and the Gauss-Newton loss.

Architecture: We are interested in learning a non-linear mapping, which maps images, $\mathbb{R}^{W\times H\times C}$ to a dense visual descriptor space, $\mathbb{R}^{W\times H\times D}$, where each pixel is represented by a D-dimensional vector. The training is performed by a Siamese encoder-decoder structured network, where we feed a pair of images, $\mathbf{I}_a$ and $\mathbf{I}_b$, producing multi-scale feature pyramids $\mathbf{F}_a^l$ and $\mathbf{F}_b^l$, where l represents the level of the decoder. For each image pair, we use a certain number of matches, denoted

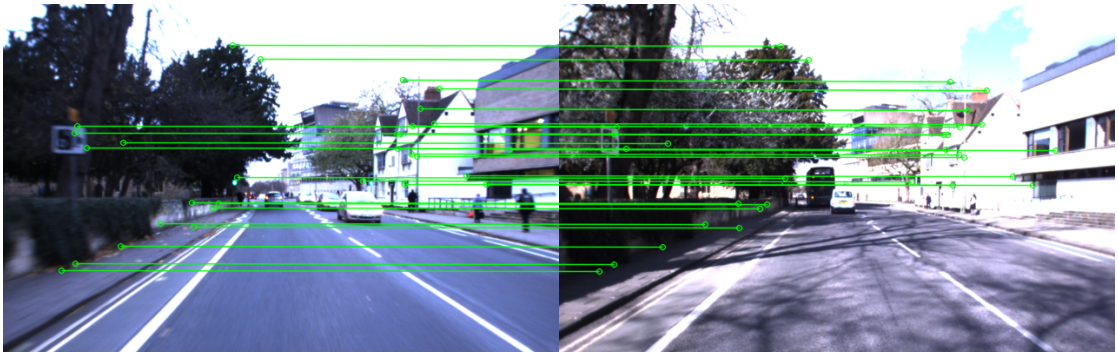


Figure 5.2: This figure shows training correspondences between a pair of images from our benchmark.

by N_{pos}, and a certain number of non-matches, denoted by N_{neg}. A pixel $\mathbf{u}_a \in \mathbb{R}^2$ from image $\mathbf{I}_a$ is considered to be a positive example if the pixel $\mathbf{u}_b \in \mathbb{R}^2$ from image $\mathbf{I}_b$ corresponds to the same 3D vertex (Figure 5.2). We make use of the inherent multi-scale hierarchy of the U-Net [97] architecture to apply the different loss terms from coarser to finer scaled pyramid levels. With this approach, our learned features will have a larger convergence radius for visual SLAM methods.

Pixelwise contrastive loss: The pixelwise contrastive loss attempts to minimize the distance between positive pairs, and maximize the distance between negative pairs. It can be computed as follows: $\mathcal{L}_{\text{contrastive}}(\mathbf{F}_a, \mathbf{F}_b, l) = \mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) + \mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l)$.

$$\mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{pos}}} \sum_{N_{\text{pos}}} D_{\text{feat}}^2 \tag{5.1}$$

$$\mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{neg}}} \sum_{N_{\text{neg}}} \max(0, M - D_{\text{feat}})^2 \tag{5.2}$$

where $D_{\text{feat}}(\cdot)$ is the L_2 distance between the feature embeddings: $D_{\text{feat}} = ||\mathbf{F}_a^l(\mathbf{u}_a) - \mathbf{F}_b^l(\mathbf{u}_b)||_2$ and M is the margin and set to 1.

Gauss-Newton algorithm for direct image alignment: Our learned deep features are ultimately applied to pose estimation. This is done using direct image alignment but generalized to a multi-channel feature map $\mathbf{F}$ with D channels. The input to this algorithm is a reference feature map $\mathbf{F}$ with known depths for some pixels in the image, and a target feature map $\mathbf{F}'$. The output is the predicted relative pose $\boldsymbol{\xi}$. Starting from an initial guess the following steps are performed iteratively:

1. All points $\mathbf{p}_i$ with known depth values are projected from the reference feature map $\mathbf{F}$ into the target feature map $\mathbf{F}'$ yielding the point $\mathbf{p'}_i$. For each of

them a residual vector $\mathbf{r} \in \mathbb{R}^D$ is computed, enforcing that the reference pixel and the target pixel should be similar:

$$\mathbf{r}_i(\mathbf{p}_i, \mathbf{p'}_i) = \mathbf{F'}(\mathbf{p'}_i) - \mathbf{F}(\mathbf{p}_i) \tag{5.3}$$

2. For each residual the derivative with respect to the relative pose is:

$$\mathbf{J}_i = \frac{d\mathbf{r}_i}{d\boldsymbol{\xi}} = \frac{d\mathbf{F'}(\mathbf{p'}_i)}{d\mathbf{p'}_i} \cdot \frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}} \tag{5.4}$$

Notice that the reference point $\mathbf{p}_i$ does not change for different solutions $\boldsymbol{\xi}$, therefore it does not appear in the derivative.

3. Using the stacked residual vector $\mathbf{r}$, the stacked Jacobian $\mathbf{J}$, and a diagonal weight matrix $\mathbf{W}$, the Gaussian system and the step $\boldsymbol{\delta}$ is computed as follows:

$$\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T\mathbf{W}\mathbf{r} \text{ and } \boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b} \tag{5.5}$$

Note that this derivation is equivalent to normal direct image alignment (as done in the frame-to-frame tracking from DSO) when replacing $\mathbf{F}$ with the image $\mathbf{I}$. In the computation of the Jacobian the numerical derivative of the features $\frac{d\mathbf{F'}(\mathbf{p'}_i)}{d\mathbf{p'}_i}$ is used. As typical images are extremely non-convex this derivative is only valid in a small vicinity (usually 1-2 pixels) around the current solution which is the main reason why direct image alignment needs a good initialization. To partially overcome this, a pyramid scheme is often used. Usually tracking on multiple channels instead of one can decrease the convergence radius ([54], [84]). However, in our case, we train the feature maps to in fact have a larger convergence basin than images by enforcing smoothness in the vicinity of the correct correspondence.

Gauss-Newton on individual pixels: Instead of running the Gauss-Newton algorithm on the 6DOF pose we can instead use it on each point $\mathbf{p}_i$ individually (which is similar to the Lucas-Kanade algorithm [98]). Compared to direct image alignment, this optimization problem has the same residual, but the parameter being optimized is the point position instead of the relative pose. In this case, the Hessian will be a 2-by-2 matrix and the step $\boldsymbol{\delta}$ can simply be added to the current pixel position (we leave out $\mathbf{W}$ for simplicity):

$$\mathbf{J'}_i = \frac{d\mathbf{F'}(\mathbf{p'}_i)}{d\mathbf{p'}_i} \text{ and } \mathbf{H'}_i = \mathbf{J'}_i^T\mathbf{J}_i \text{ and } \mathbf{b'}_i = \mathbf{J'}_i^T\mathbf{r}_i \tag{5.6}$$

These individual Gauss-Newton systems can be combined with the system for 6DOF pose estimation (Equation (5.5)) using:
$\mathbf{H} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{H'}_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)$ and $\mathbf{b} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{b'}_i$ The difference between our simplified systems and the one for pose estimation is only the derivative with respect to the pose, which is much smoother than the image derivative [15].

This means that if the Gauss-Newton algorithm performs well on individual pixels it will also work well on estimating the full pose. Therefore, we propose to train a neural network on correspondences which are easy to obtain, *e.g.* using a SLAM method, and then later apply it for pose estimation.

We argue that training on these individual points is superior to training on the 6DOF pose. The estimated pose can be correct even if some points contribute very wrong estimates. This increases robustness at runtime but when training we want to improve the information each point provides. Also, when training on the 6DOF pose we only have one supervision signal for each image pair, whereas when training on correspondences we have over a thousand signals. Hence, our method exhibits exceptional generalization capabilities as shown in the results section.

The probabilistic Gauss-Newton loss: The linear system described in Equation (5.6) defines a 2-dimensional Gaussian probability distribution. The reason is that the Gauss-Newton algorithm tries to find the solution with maximum probability in a least squares fashion. This can be derived using the negative log-likelihood of the Gaussian distribution:

$$E(\mathbf{x}) = -\log f_X(\mathbf{x}) = \tag{5.7}$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) = \tag{5.8}$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{H}(\mathbf{x}-\boldsymbol{\mu}) + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|) \tag{5.9}$$

where $\mathbf{x}$ is a pixel position and $\boldsymbol{\mu}$ is the mean.

In the Gauss-Newton algorithm the mean (which also corresponds to the point with maximum probability) is computed with $\boldsymbol{\mu} = \mathbf{x}_s + \boldsymbol{\delta}$, where the $\boldsymbol{\delta}$ comes from Equation (5.5) and $\mathbf{x}_s$ denotes the start point. To derive this, only the first term is used (because the latter parts are constant for all solutions $\mathbf{x}$). In our case, however, the second term is very relevant, because the network can influence both $\boldsymbol{\mu}$ and $\mathbf{H}$.

This derivation shows, that $\mathbf{H}, \mathbf{b}$ as computed in the GN-algorithm, also define a Gaussian probability distribution with mean $\mathbf{x}_s + \mathbf{H}^{-1}\mathbf{b}$ and covariance $\mathbf{H}^{-1}$.

When starting with an initial solution $\mathbf{x}_s$ the network should assign maximal probability to the pixel that marks the correct correspondence. With $\mathbf{x}$ being the correct correspondence, we therefore use $E(\mathbf{x}) =$ Equation (5.9) as our loss function which we call the *Gauss-Newton loss* (see Algorithm 1).

In the algorithm, a small number ϵ is added to the diagonal of the Hessian, to ensure it is invertible.

Analysis of the Gauss-Newton loss: By minimizing Equation (5.9) the network has to maximize the probability density of the correct solution. As the integral over the probability densities always has to be 1, the network has the choice to either focus all the density on a small set of solutions (with more risk of being penalized if this solution is wrong), or to distribute the density to more solutions

Algorithm 1 Compute Gauss-Newton loss

$\mathbf{F}_a \leftarrow \text{network}(\mathbf{I}_a)$
$\mathbf{F}_b \leftarrow \text{network}(\mathbf{I}_b)$
$e \leftarrow 0$ ▷ Total error
for all correspondences $\mathbf{u}_a, \mathbf{u}_b$ **do**
  $\mathbf{f}_t \leftarrow \mathbf{F}_a(\mathbf{u}_a)$ ▷ Target feature
  $\mathbf{x}_s \leftarrow \mathbf{u}_b + \text{rand}(\text{vicinity})$ ▷ Compute start point
  $\mathbf{f}_s \leftarrow \mathbf{F}_b(\mathbf{x}_s)$
  $\mathbf{r} \leftarrow \mathbf{f}_s - \mathbf{f}_t$ ▷ Residual
  $\mathbf{J} \leftarrow \frac{d\mathbf{F}_b}{d\mathbf{x}_s}$ ▷ Numerical derivative
  $\mathbf{H} \leftarrow \mathbf{J}^T\mathbf{J} + \epsilon \cdot \text{Id}$ ▷ Added epsilon for invertibility
  $\mathbf{b} \leftarrow \mathbf{J}^T r$
  $\boldsymbol{\mu} \leftarrow \mathbf{x}_s - \mathbf{H}^{-1}\mathbf{b}$
  $e_1 \leftarrow \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$ ▷ First error term
  $e_2 \leftarrow \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$ ▷ Second error term
  $e \leftarrow e + e_1 + e_2$
end for

which in turn will have a lower individual density. By maximizing the probability of the correct solution, the network is incentivized to improve the estimated solution and its certainty.

This is also reflected in the two parts of the loss. The first term $e_1 = \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$ penalizes deviations between the estimated and the correct solution, scaled with the Hessian $\mathbf{H}$. The second term $e_2 = \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$ is large if the network does not output enough certainty for its solution. This means that the network can reduce the first error term e_1 by making $\mathbf{H}$ smaller. As a consequence, the second error term will be increased, as this will also reduce the determinant of $\mathbf{H}$. Notice also that this can be done in both dimensions independently. The network has the ability to output a large uncertainty in one direction, but a small uncertainty in the other direction. This is one of the traditional advantages of direct methods which are naturally able to utilize also lines instead of just feature points.

From Equation (5.9) it can be observed that the predicted uncertainty depends only on the numerical derivative of the target image at the start position. The higher the gradients the higher the predicted certainty. In DSO this is an unwanted effect that is counteracted by the gradient-dependent weighting applied to the cost-function [15, Equation (7)]. In our case, however, it gives the network the possibility to express its certainty and incentivizes it to output discriminative features.

Upon training the network with our loss formulation, we observe that the features are very similar despite being generated from images taken from sequences

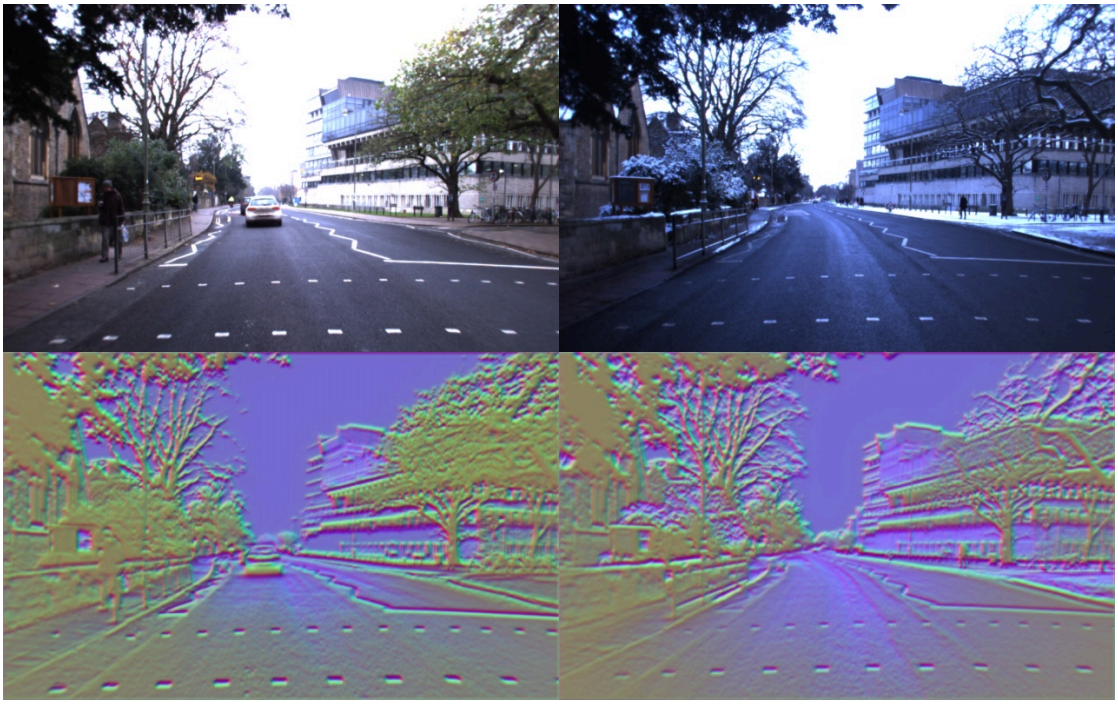


Figure 5.3: This figure shows images and their corresponding feature maps predicted by our GN-Net for the Oxford RobotCar dataset. Each column depicts the image and feature map for a sample taken from 2 different sequences. Despite lighting and weather changes, the feature maps are robust to these variations. The visualization of the features shows the high-dimensional descriptors reduced to 3D through PCA.

with different lighting/weather conditions, as shown in Figure 5.3.

5.4 Relocalization Tracking Benchmark

Previous tasks for localization/odometry can primarily be divided into two categories:

- Odometry datasets [19], [70], where there is a continuous stream of images (sometimes combined with additional sensor data like IMUs).
- Image collections where individual images are usually further apart from each other in space/time [28], [99].

We argue that for several applications a combination of these two tasks which we refer to as *relocalization tracking* is a more realistic scenario. The idea is that the algorithm has two inputs:

1. An image sequence (like a normal odometry dataset).
2. A collection of individual images (possibly with different weathers/times), each of which shall be tracked against one specific image from point 1.

The algorithm is supposed to track the normal sequential image sequence and at the same time perform tracking of the images in point 2. The advantage of this task is that the used algorithm can utilize the temporally continuous sequence from point 1 to compute accurate depth values for a part of the image (using a standard visual odometry method), which can then be used to improve the tracking of the individual images of point 2.

This task is very realistic as it comes up when tracking an image sequence and at the same time trying to relocalize this sequence in a prior map. A similar challenge occurs by trying to merge multiple maps from different times. In both cases, one has more information than just a random collection of images. It is important to reiterate here that the task of finding relocalization candidates is not considered but rather tracking them with maximum accuracy/robustness is the focus. This is because our benchmark decouples image retrieval from tracking.

We have created a benchmark for *relocalization tracking* using the CARLA simulator and the Oxford RobotCar dataset. Our benchmark includes ground-truth poses between different sequences for both training, validation, and testing.

CARLA: For synthetic evaluations, we use CARLA version 0.8.2. We collect data for 3 different weather conditions representing *WetNoon*, *SoftRainNoon*, and *WetCloudySunset*. We recorded the images at a fixed framerate of 10 frames per second (FPS). At each time step, we record images and its corresponding dense depth map from 6 different cameras with different poses rendered from the simulation engine, which means that the poses in the benchmark are not limited to just 2DOF. The images and the dense depth maps are of size 512×512. For each weather condition, we collected 3 different sequences comprising 500-time steps with an average distance of 1.6m. This is done for training, validation, and testing, meaning there are 27 sequences, containing 6 cameras each. Training, validation, and test sequences were all recorded in different parts of the CARLA town. We have generated the test sequences after all hyperparameter tuning of our method was finished, meaning we had no access to the test data when developing the method. In accordance, we shall withhold the ground-truth for the test sequences.

Oxford RobotCar: Creating a multi-weather benchmark for this dataset imposes various challenges because the GPS-based ground-truth is very inaccurate. To find the relative poses between images from different sequences we have used the following approach. For pairs of images from two different sequences, we accumulate the point cloud captured by the 2D lidar for 60 meters using the visual odometry result provided by the Oxford dataset. The resulting two point clouds are aligned with the global registration followed by ICP alignment using the implementation of Open3D [100]. We provide the first pair of images manually and the following

pairs are found using the previous solution. We have performed this alignment for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*. The average relocalization distance across all sequences is 0.84m.

5.5 Experimental Evaluation

We perform our experiments on the *relocalization tracking* benchmark described in Section 5.4. We demonstrate the multi-weather relocalization performance on both the CARLA and the Oxford RobotCar dataset. For the latter, we show that our method even generalizes well to unseen weather conditions like rain or snow while being trained only on the sunny and overcast conditions. Furthermore, a qualitative relocalization demo[1] on the Oxford RobotCar dataset is provided, where we demonstrate that our GN-Net can facilitate precise relocalization between weather conditions.

We train our method using sparse depths created by running Stereo DSO on the training sequences. We use intra-sequence correspondences calculated using the DSO depths and the DSO pose. Meanwhile, inter-sequence correspondences are obtained using DSO depths and the ground-truth poses provided by our benchmark. The ground truth poses are obtained via Lidar alignment for Oxford and directly from the simulation engine for CARLA as explained in Section 5.4. Training is done from scratch with randomly initialized weights and an ADAM optimizer with a learning rate of 10^{-6}. The image pair fed to the Siamese network is randomly selected from any of the training sequences while ensuring that the images in the pair do not differ by more than 5 keyframes. Each branch of the Siamese network is a modified U-Net architecture with shared weights. Further details of the architecture and training can be found in the supplementary material[1]. Note that at inference time, only one image is needed to extract the deep visual descriptors, used as input to the SLAM algorithm. While in principle, our approach can be deployed in conjunction with any direct method, we have coupled our deep features with Direct Sparse Odometry (DSO).

We compare to state-of-the-art **direct** methods:

Stereo Direct Sparse Odometry (DSO) [101]: Whenever there is a relocalization candidate for a frame we ensure that the system creates the corresponding keyframe. This candidate is tracked using the *coarse tracker*, performing direct image alignment in a pyramid scheme. We use the identity as initialization without any other random guesses for the pose.

[1] https://vision.in.tum.de/gn-net.

GN-Net (Ours): Same as with DSO, however, for relocalization tracking, we replace the grayscale images with features created by our GN-Net on all levels of the feature pyramid. The network is trained with the Gauss-Newton loss formulation described in Section 5.3.

We also compare to state-of-the-art **indirect** methods:

ORB-SLAM [31]: For relocalization tracking, we use the standard feature-based 2-frame pose optimization also used for frame-to-keyframe tracking. We have also tried the RANSAC scheme implemented in ORB-SLAM for relocalization, however, it yielded worse results overall. Thus we will report only the default results.

D2-Net [102], SuperPoint [103]: For both methods we use the models provided by the authors. The relative pose is estimated using the OpenCV implementation of the PnP algorithm in a RANSAC scheme.

We also evaluated the Deeper Inverse Compositional Algorithm [92] on the *relocalization tracking* benchmark. However, the original implementation didn't converge despite multiple training trials with different hyperparameters.

For all our quantitative experiments we plot a cumulative distribution of the relocalization error, which is the norm of the translation between the estimated and the correct solution in meters. For each relocalization error between 0 and 1 meter, it plots the percentage of relocalization candidates that have been tracked with at least this accuracy.

5.5.1 Quantitative multi-weather evaluation

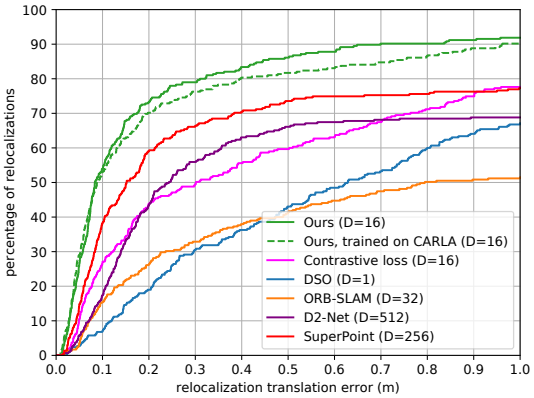
We demonstrate the relocalization tracking accuracy on our new benchmark across different weathers. For these experiments, tracking is performed only across sequences with a different weather condition.

CARLA: For this experiment, we train on the training sequences provided by our benchmark. For all learning-based approaches, the best epoch is selected using the relocalization tracking performance on the validation set. The results on the test data are shown in the supplementary[2].

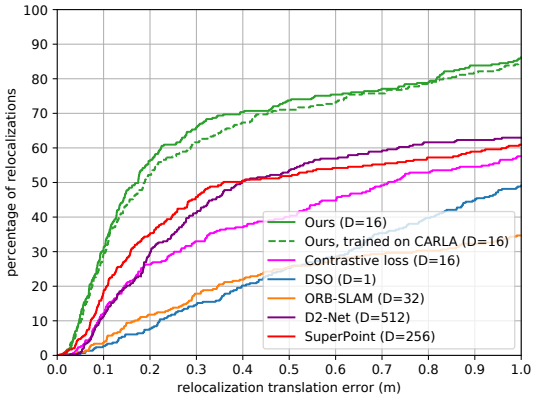
Oxford RobotCar: We train on the sunny and overcast condition correspondences provided by our *relocalization tracking* benchmark for the Oxford dataset. For the learning-based methods, we select the best epoch based on the relocalization tracking performance on the training set. We use the same hyperparameters that were found using the CARLA validation set. We show the results on the test data in Figure 5.4. Our method significantly outperforms the baselines. The Gauss-Newton loss has a large impact as compared to the model trained with only the contrastive loss.

Figures 5.4b-f show how well our model generalizes to unseen weather conditions. Despite being trained only on two sequences with overcast and sunny conditions

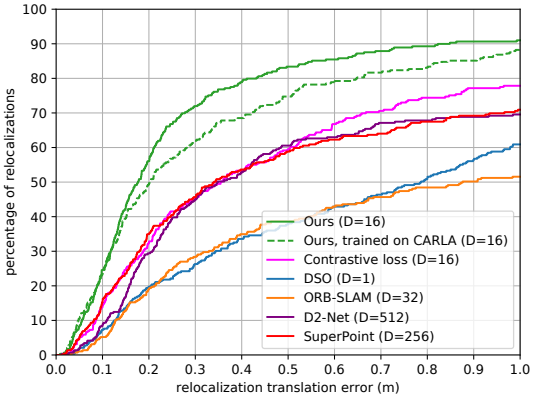
[2] https://vision.in.tum.de/gn-net.



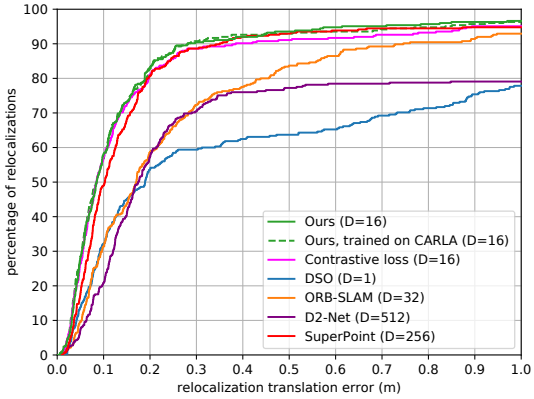
(a) Relocalization sunny and overcast.



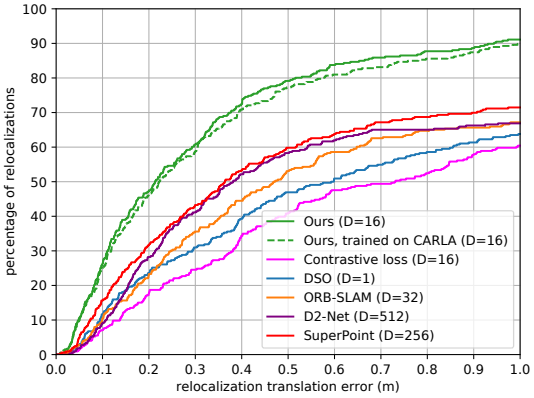
(b) Relocalization sunny and rainy.



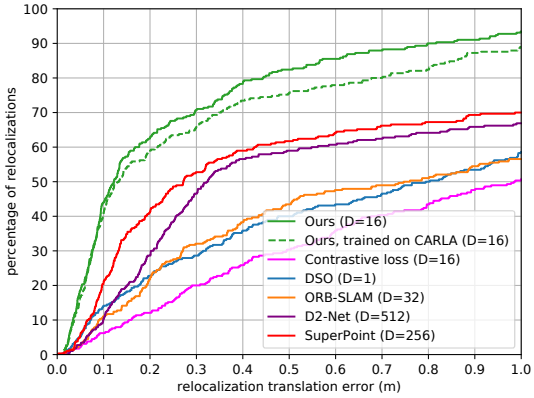
(c) Relocalization sunny and snowy.



(d) Relocalization overcast and rainy.



(e) Relocalization overcast and snowy.



(f) Relocalization rainy and snowy.

Figure 5.4: This figure shows the cumulative relocalization accuracy on the Oxford RobotCar dataset for different sequences. D denotes the dimension of the feature descriptor. Our method achieves the highest accuracy across all sequences. It is interesting to observe that despite being trained only on two sequences in overcast and sunny condition, our model still generalizes very well to even *unseen* rainy and snowy conditions. Even the model trained only on the synthetic CARLA benchmark outperforms all baselines, showing exceptional generalization capabilities.

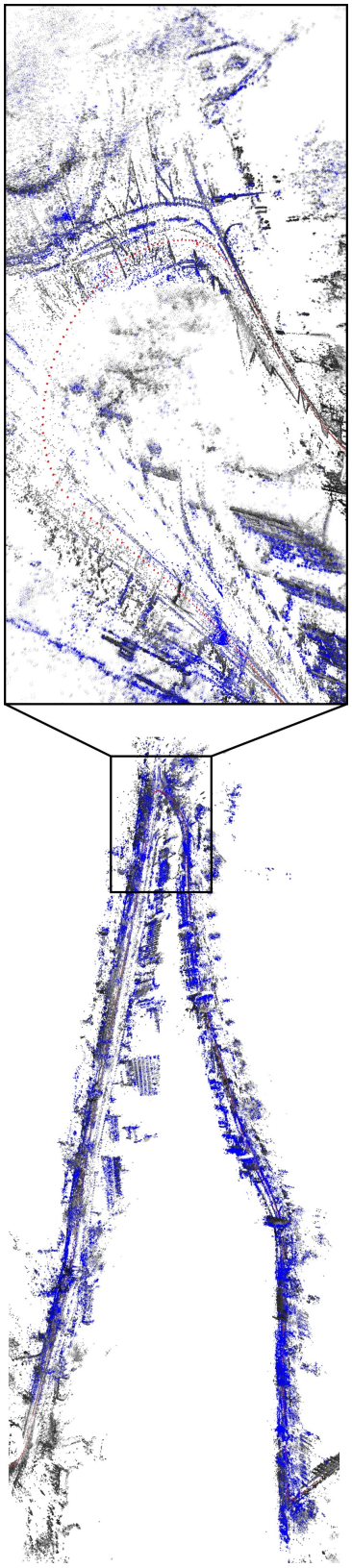


Figure 5.5: This figure shows a point cloud result of our GN-Net. We relocalize a rainy sequence (blue) against a reference map created from the sunny sequence (gray).

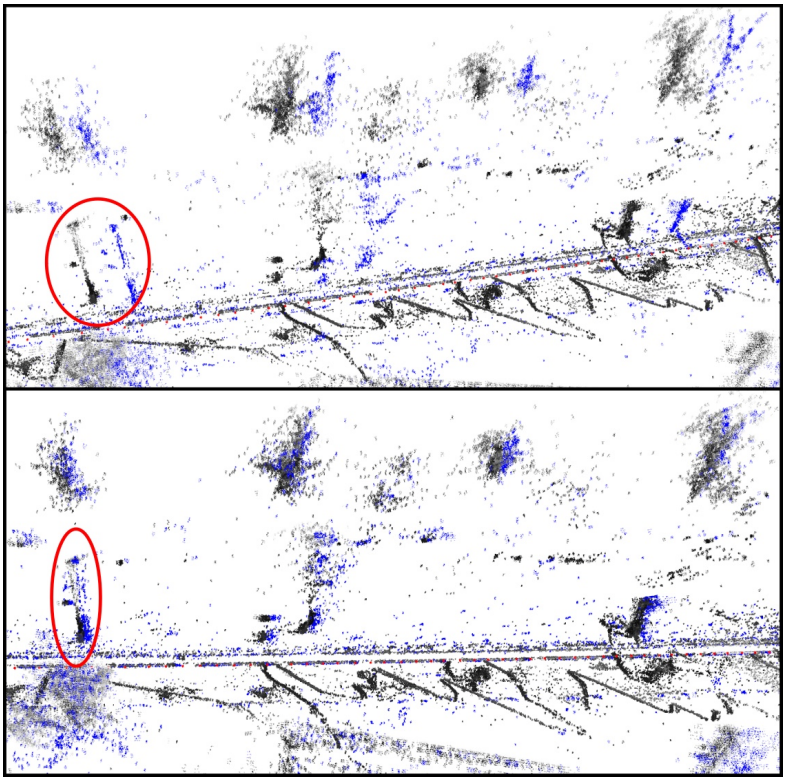


Figure 5.6: Top: relocalization using the model trained with only the contrastive loss. Bottom: relocalization using the model trained with our loss formulation. This visually demonstrates the influence of the Gauss-Newton loss.

the results for tracking against a rainy and a snowy sequence are almost the same. Interestingly our model which was trained only on the CARLA benchmark outperforms all baselines significantly.

5.5.2 Qualitative multi-weather evaluation

Finally, we show a relocalization demo comparing our GN-Net to DSO. For this, we load a point cloud from a sequence recorded in the sunny condition and relocalize against sequences from rainy and snowy conditions. For each keyframe, we try to track it against the nearest keyframe in the map according to the currently estimated transformation between the trajectory and the map. Figure 5.5 shows that the point clouds from the different sequences align nicely, despite belonging to different weather conditions. This experiment shows that our method can perform the desired operations successfully on a real-world application, including relocalization against unseen weather conditions. Figure 5.6 demonstrates the difference between our Gauss-Newton loss and the contrastive loss. This shows that the quantitative improvement has a visible effect on the application of relocalization.

Figure 5.7: shows image pairs used in the qualitative relocalizations. Left: rainy (top row) and snowy (bottom row) images relocalized against the sunny reference images (right).

Figure 5.7 shows sample images used in the qualitative relocalizations.

5.5.3 Additional experiments on EuRoC and CARLA

In the supplementary, we provide more evaluations on datasets with and without brightness variations. This includes relocalization tracking on the CARLA benchmark and visual odometry on the EuRoC [70] dataset. We show that also in these situations our deep features significantly outperform DSO and ORB-SLAM because of their robustness to large-baselines. On the EuRoC dataset, we improve the DSO performance by almost a factor of 2 for low-framerates.

5.6 Conclusion & Future Work

With the advent of deep learning, we can devise feature space encodings that are designed to be optimally suited for the subsequent visual SLAM algorithms. More specifically, we propose to exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm to devise a novel loss function for feature space encoding that we call the Gauss-Newton loss. It is designed to promote

robustness to strong lighting and weather changes while enforcing a maximal basin of convergence for the respective SLAM algorithm. Quantitative experiments on synthetic and real-world data demonstrates that the Gauss-Newton loss allows us to significantly expand the realm of applicability of direct visual SLAM methods, enabling relocalization and map merging across drastic variations in weather and illumination.

Chapter 6

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization

Authors	Lukas von Stumberg[1,2*]	stumberg@in.tum.de
	Patrick Wenzel[1,2*]	wenzel@in.tum.de
	Nan Yang[1,2]	yangn@in.tum.de
	Daniel Cremers[1,2]	cremers@in.tum.de

[*]These authors contributed equally

[1]Technical University of Munich, Munich, Germany
[2]Artisense

Publication

L. von Stumberg, P. Wenzel, N. Yang, and D. Cremers, **LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization**, in *International Conference on 3D Vision (3DV)*, 2020, pp. 968–977. DOI: 10.1109/3DV50981.2020.00107. arXiv: 2010.06323

Contribution		
	Problem definition	*significantly contributed*
	Literature survey	*significantly contributed*
	Algorithm development	*significantly contributed*
	Method implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

Abstract We present LM-Reloc – a novel approach for visual relocalization based on direct image alignment. In contrast to prior works that tackle the problem with a feature-based formulation, the proposed method does not rely on feature matching and RANSAC. Hence, the method can utilize not only corners but any region of the image with gradients. In particular, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. The learned features significantly improve the robustness of direct image alignment, especially for relocalization across different conditions. To further improve the robustness of LM-Net against large image baselines, we propose a pose estimation network, CorrPoseNet, which regresses the relative pose to bootstrap the direct image alignment. Evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark show that our approach delivers more accurate results than previous state-of-the-art methods while being comparable in terms of robustness.

6.1 Introduction

Map-based relocalization, that is, to localize a camera within a pre-built reference map, is becoming more and more important for robotics [94], autonomous driving [104], [105] and AR/VR [106]. Sequential-based approaches, which leverage the temporal structure of the scene provide more stable pose estimations and also deliver the positions in global coordinates compared to single image-based localization methods. The map is usually generated by either using LiDAR or visual Simultaneous Localization and Mapping (vSLAM) solutions. In this paper, we consider vSLAM maps due to the lower-cost visual sensors and the richer semantic information from the images. Feature-based methods [24], [25], [29], [31] and direct methods [15], [22], [32], [47] are two main lines of research for vSLAM.

Once a map is available, the problem of relocalizing within this map at any later point in time requires to deal with long-term changes in the environment. This makes a centimeter-accurate global localization challenging, especially in the presence of drastic lighting and appearance changes in the scene. For this task, feature-based methods are the most commonly used approaches to estimate the ego-pose and its orientation. This is mainly due to the advantage that features are more robust against changes in lighting/illumination in the scene.

However, feature-based methods can only utilize keypoints that have to be matched across the images before the pose estimation begins. Thus they ignore large parts of the available information. Direct methods, in contrast, can take advantage of all image regions with sufficient gradients and as a result, are known to be more accurate on visual odometry benchmarks [15], [107], [108].

In this paper, we propose LM-Reloc, which applies direct techniques to the

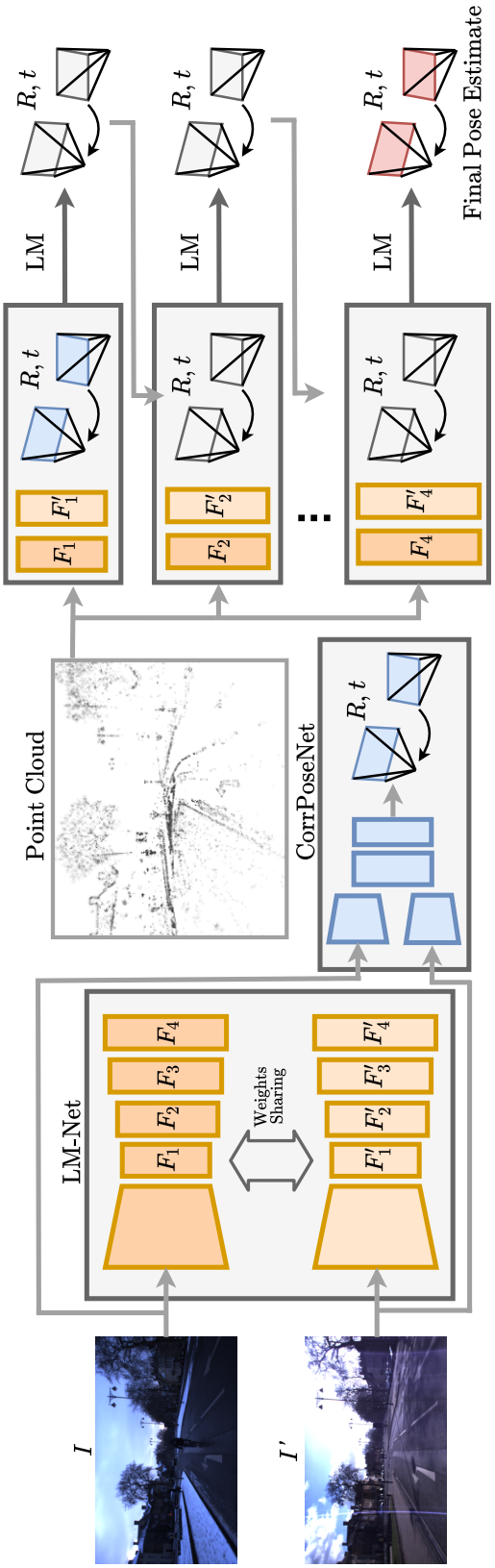


Figure 6.1: We propose LM-Reloc – a novel approach for visual relocalization based on direct image alignment. It consists of two deep neural networks: LM-Net, an encoder-decoder network for learning dense visual descriptors and a CorrPoseNet to bootstrap the direct image alignment. The final 6DoF relative pose estimate between image I and I' is obtained in a coarse-to-fine pyramid scheme leveraging the learned feature maps. The initialization for the direct image alignment is obtained by the CorrPoseNet.

task of relocalization. LM-Reloc consists of LM-Net, CorrPoseNet, and a non-linear optimizer, which work seamlessly together to deliver reliable pose estimation without RANSAC and feature matching. In particular, we derive a loss formulation, which is specifically designed to work well with the Levenberg-Marquardt (LM) algorithm [63], [64]. We use a deep neural network, LM-Net, to train descriptors that are being fed to the direct image alignment algorithm. Using these features results in better robustness against bad initializations, large baselines, and against illumination changes.

While the robustness improvements gained with our loss formulation are sufficient in many cases, for very large baselines or strong rotations, some initialization can still be necessary. To this end, we propose a pose estimation network. Based on two images it directly regresses the 6DoF pose, which we utilize as initialization for LM-Net. The CorrPoseNet contains a correlation layer as proposed in [109], which ensures that the network can handle large displacements. The proposed CorrPoseNet displays a lot of synergies with LM-Net. Despite being quite robust, the predictions of the CorrPoseNet are not very accurate. Thus it is best used in conjunction with our LM-Net, resulting in very robust and accurate pose estimates.

We evaluate our approach on the relocalization tracking benchmark from [5], which contains scenes simulated using CARLA [81], as well as sequences from the Oxford RobotCar dataset [82]. Our LM-Net shows superior accuracy especially in terms of rotation while being competitive in terms of robustness.

We summarize our main contributions:

- LM-Reloc, a novel pipeline for visual relocalization based on direct image alignment, which consists of LM-Net, CorrPoseNet, and a non-linear optimizer.
- A novel loss formulation together with a point sampling strategy that is used to train LM-Net such that the resulting feature descriptors are optimally suited to work with the LM algorithm.
- Extensive evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark which show that the proposed approach achieves state-of-the-art relocalization accuracy without relying on feature matching or RANSAC.

6.2 Related Work

In this section, we review the main topics that are closely related to our work, including direct methods for visual localization and feature-based visual localization methods.

Direct methods for visual localization. In recent years, direct methods [15], [22], [32] for SLAM and visual odometry have seen a great progress. Unlike feature-based methods [24], [25], [29], [31] which firstly extracts keypoints as well as the corresponding descriptors, and then minimize the geometric errors, direct methods minimize the energy function based on the photometric constancy assumption without performing feature matching or RANSAC. By utilizing more points from the images, direct methods show higher accuracy than feature-based methods [108]. However, classical direct methods show lower robustness than feature-based methods when the photometric constancy assumption is violated due to, e.g., the lighting and weather changes which are typical for long-term localization [28]. In [84] and [110], the authors propose to use the handcrafted features to improve the robustness of direct methods against low light or global appearance changes. Some recent works [5], [89], [92] address the issue by using learned features from deep neural networks [111]. In [89] they train deep features using a Hinge-Loss based on the Lucas-Kanade method, however, in contrast to us, they estimate the optical flow instead of applying the features to the task of relocalization. The most related work to ours is GN-Net [5] which proposes a Gauss-Newton loss to learn deep features. By performing direct image alignment on the learned features, GN-Net can deliver reliable pose estimation between the images taken from different weather or season conditions. The proposed LM-Net further derives the loss formulation based on Levenberg-Marquardt to improve the robustness against bad initialization compared to the Gauss-Newton method. Inspired by D3VO [8], LM-Reloc also proposes a relative pose estimation network with a correlation layer [109] to regress a pose estimate which is used as the initialization for the optimization.

Feature-based visual localization. Most approaches for relocalization utilize feature detectors and descriptors, which can either be handcrafted, such as SIFT [112] or ORB [113], or especially in the context of drastic lighting and appearance changes can be learned. Recently, many descriptor learning methods have been proposed which follow a *detect-and-describe* paradigm, e.g., SuperPoint [103], D2-Net [102], or R2D2 [114]. Moreover, SuperGlue [115], a learning-based alternative to the matching step of feature-based methods has been proposed and yields significant performance improvements. For a complete relocalization pipeline the local pose refinement part has to be preceded by finding the closest image in a database given a query [80]. While some approaches [116]–[118] address the joint problem, in this work, we decouple these two tasks and only focus on the pose refinement part.

6.3 Method

In this work, we address the problem of computing the 6DoF pose $\boldsymbol{\xi} \in SE(3)$ between two given images I and I'. Furthermore, we assume that depths for a sparse set of points P are available, e.g., by running a direct visual SLAM system

such as DSO [15].

The overall pipeline of our approach is shown in Figure 6.1. It is composed of LM-Net, CorrPoseNet, and a non-linear optimizer using the LM algorithm. LM-Net is trained with a novel loss formulation designed to learn feature descriptors optimally suited for the LM algorithm. The encoder-decoder architecture takes as input a reference image I as well as a target image I'. The network is trained end-to-end and will produce multi-scale feature maps F_l and F'_l, where $l = 1, 2, 3, 4$ denotes the different levels of the feature pyramid. In order to obtain an initial pose estimate for the non-linear optimization, we propose CorrPoseNet, which takes I and I' as the inputs and regress their relative pose. Finally, the multi-scale feature maps together with the depths obtained from DSO [15] form the non-linear energy function which is minimized using LM algorithm in a coarse-to-fine manner to obtain the final relative pose estimate. In the following, we will describe the individual components of our approach in more detail.

6.3.1 Direct Image Alignment with Levenberg-Marquardt

In order to optimize the pose $\boldsymbol{\xi}$ (consisting of rotation matrix $\mathbf{R}$ and translation $\mathbf{t}$), we minimize the feature-metric error:

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{p} \in P} \left\| F'_l(\mathbf{p}') - F_l(\mathbf{p}) \right\|_\gamma , \tag{6.1}$$

where $||\cdot||_\gamma$ is the Huber norm and $\mathbf{p}'$ is the point projected onto the target image I' using the depths and the pose:

$$\mathbf{p}' = \Pi \left(\mathbf{R}\, \Pi^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t} \right). \tag{6.2}$$

This energy function is first minimized on the coarsest pyramid level 1, whose feature maps F_1 have a size of $(w/8, h/8)$, yielding a rough pose estimate. The estimate is refined by further minimizing the energy function on the subsequent pyramid levels 2, 3, and 4, where F_4 has the size of the original image (w, h). In the following, we provide details of the minimization performed in every level and for simplicity we will denote F_l as F from now on.

Minimization is performed using the Levenberg-Marquardt algorithm. In each iteration we compute the update $\boldsymbol{\delta} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$ as follows: Using the residual vector $\mathbf{r} \in \mathbb{R}^n$, the Huber weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, and the Jacobian of the residual vector with respect to the pose $\mathbf{J} \in \mathbb{R}^{n \times 6}$, we compute the Gauss-Newton system:

$$\mathbf{H} = \mathbf{J}^T \mathbf{W} \mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T \mathbf{W} \mathbf{r}. \tag{6.3}$$

The damped system can be obtained with either Levenberg's formula [63]:

$$\mathbf{H}' = \mathbf{H} + \lambda \mathbf{I} \tag{6.4}$$

or the Marquardt's formula [64]:

$$\mathbf{H}' = \mathbf{H} + \lambda \operatorname{diag}(\mathbf{H}) \tag{6.5}$$

depending on the specific application.

The update $\boldsymbol{\delta}$ and the pose $\boldsymbol{\xi}^i$ in the iteration i are computed as:

$$\boldsymbol{\delta} = {\mathbf{H}'}^{-1}\mathbf{b} \text{ and } \boldsymbol{\xi}^i = \boldsymbol{\delta} \boxplus \boldsymbol{\xi}^{i-1}, \tag{6.6}$$

where $\boxplus : \mathfrak{se}(3) \times SE(3) \to SE(3)$ is defined as in [15].

The parameter λ can be seen as an interpolation factor between gradient descent and the Gauss-Newton algorithm. When λ is high the method behaves like gradient descent with a small step size, and when it is low it is equivalent to the Gauss-Newton algorithm. In practice, we start with a relatively large λ and multiply it by 0.5 after a successful iteration, and by 4 after a failed iteration [15].

Figure 6.2 shows the typical behaviour of the algorithm. In the beginning the initial pose is inaccurate, resulting in projected point positions, which are a couple of pixels away from the correct location. λ will be high meaning that the algorithm will behave similar to gradient descent. After a couple of iterations, the pose got more accurate, and the projected points are in a closer vicinity to the correct location. By now, λ has probably decreased, so the algorithm will behave more similar to the Gauss-Newton algorithm. Now we expect the algorithm to converge quickly.

6.3.2 Loss Formulation for Levenberg-Marquardt

The key contribution of this work is LM-Net which provides feature maps F that improve the convergence behaviour of the LM algorithm and, in the meantime, are invariant to different conditions. We train our network in a Siamese fashion based on ground-truth pixel correspondences.

In this section, $\mathbf{p}$ denotes a reference point (located on image I) and the ground-truth correspondence (located on image I') is $\mathbf{p}'_{\text{gt}}$. For the loss functions explained below we further categorize $\mathbf{p}'$ into $\mathbf{p}'_{\text{neg}}$, $\mathbf{p}'_{\nabla}$, and $\mathbf{p}'_{\nabla^2}$, which is realized by using different negative correspondence sampling. Our loss formulation is inspired by the typical behaviour of the Levenberg-Marquardt algorithm explained in the previous section (see Figure 6.2). For a point, we distinguish four cases which can happen during the optimization:

1. The point is at the correct location ($\mathbf{p}'_{\text{gt}}$).
2. The point is an outlier ($\mathbf{p}'_{\text{neg}}$).
3. The point is relatively far from the correct solution ($\mathbf{p}'_{\nabla}$).

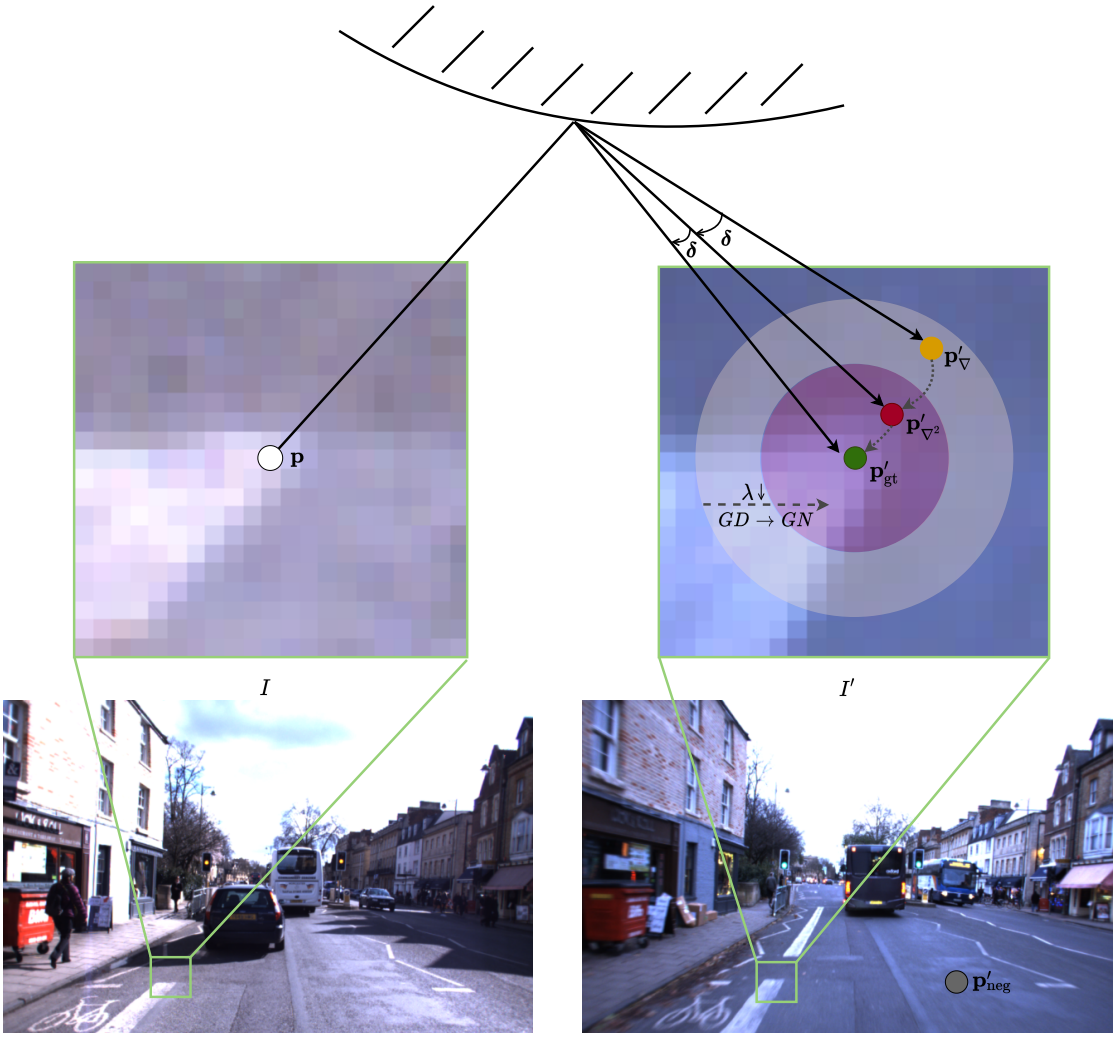


Figure 6.2: Visualization of the typical behavior of direct image alignment with Levenberg-Marquardt. Initially, the projected point position (orange point, $\mathbf{p}'_{\nabla}$) is far away from the correct solution (green point, $\mathbf{p}'_{\text{gt}}$), and λ is large, yielding an update step similar to gradient descent. After some iterations the projected point position gets closer to the optimum (red point, $\mathbf{p}'_{\nabla^2}$) and at the same time λ will get smaller, leading to an update step similar to the Gauss-Newton algorithm. This is the intuition behind our point sampling strategy, where we utilize the ground-truth correspondence $\mathbf{p}'_{\text{gt}}$ for Equation (6.7), a negative $\mathbf{p}'_{\text{neg}}$ sampled across the whole image for Equation (6.8), a negative $\mathbf{p}'_{\nabla}$ sampled in a far vicinity for Equation (6.12), and a negative $\mathbf{p}'_{\nabla^2}$ sampled in a close vicinity for Equation (6.14).

4. The point is very close to the correct solution ($\mathbf{p}'_{\nabla^2}$).

In the following we will derive a loss function for each of the 4 cases:

1. The point is already at the correct location. In this case we would like the residual to be as small as possible, in the best case 0.

$$E_{\text{pos}} = \|F'(\mathbf{p}'_{\text{gt}}) - F(\mathbf{p})\|^2 \tag{6.7}$$

2. The point is an outlier or the pose estimate is completely wrong. In this case the projected point position can be at a completely different location than the correct correspondence. In this scenario we would like the residual of this pixel to be very large to reflect this, and potentially reject a wrong update. To enforce this property we sample a negative correspondences $\mathbf{p}'_{\text{neg}}$ uniformly across the whole image, and compute

$$E_{\text{neg}} = \max\left(M - \|F'(\mathbf{p}'_{\text{neg}}) - F(\mathbf{p})\|^2, 0\right) \tag{6.8}$$

where M is the margin how large we would like the energy of a wrong correspondence to be. In practice, we set it to 1.

3. The predicted pose is relatively far away from the optimum, meaning that the projected point position will be a couple of pixels away from the correct location. As this typically happens during the beginning of the optimization we assume that λ will be relatively large and the algorithm behaves similar to gradient descent. In this case we want that the gradient of this point is oriented in the direction of the correct solution, so that the point has a positive influence on the update step.

For computing a loss function to enforce this property we sample a random negative correspondence $\mathbf{p}'_{\nabla}$ in a relatively large vicinity around the correct solution (in our experiments we use 5 pixels distance). Starting from this negative correspondence $\mathbf{p}'_{\nabla}$ we first compute the 2×2 Gauss-Newton system for this individual point, similarly to how it is done for optical flow estimation using Lucas-Kanade:

$$\mathbf{r}_{\mathbf{p}}(\mathbf{p}, \mathbf{p}'_{\nabla}) = \mathbf{F}'(\mathbf{p}'_{\nabla}) - \mathbf{F}(\mathbf{p}) \tag{6.9}$$

$$\mathbf{J}_{\mathbf{p}} = \frac{d\mathbf{F}'(\mathbf{p}'_{\nabla})}{d\mathbf{p}'_{\nabla}} \text{ and } \mathbf{H}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T\mathbf{J}_{\mathbf{p}} \text{ and } \mathbf{b}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T\mathbf{r}_{\mathbf{p}} \tag{6.10}$$

We compute the damped system using a relatively large fixed λ_f, as well as the optical flow step[1]

$$\mathbf{H}'_{\mathbf{p}} = \mathbf{H}_{\mathbf{p}} + \lambda_f \mathbf{I} \text{ and } \mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla} + \mathbf{H}'^{-1}_{\mathbf{p}}\mathbf{b}_{\mathbf{p}}. \tag{6.11}$$

[1]Here we use Equation (6.4) instead of Equation (6.5) since we find it more stable for training LM-Net.

In order for this point to have a useful contribution to the direct image alignment, this update step should move in the correct direction by at least δ. We enforce this using a Gradient-Descent loss function which is small only if the distance to the correct correspondence *after* the update is smaller than before the update:

$$E_{\text{GD}} = \max\left(\|\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\|^2 - \|\mathbf{p}'_{\nabla} - \mathbf{p}'_{\text{gt}}\|^2 + \delta, 0\right) \tag{6.12}$$

In practice, we choose $\lambda_f = 2.0$ and $\delta = 0.1$.

4. The predicted pose is very close to the optimum, yielding a projected point position in very close proximity of the correct correspondence, and typically λ will be very small, so the update will mostly be a Gauss-Newton step. In this case we would like the algorithm to converge as quickly as possible, with subpixel accuracy. We enforce this using the Gauss-Newton loss [5]. To compute it we first sample a random negative correspondence $\mathbf{p}'_{\nabla^2}$ in a 1-pixel vicinity around the correct location. Then we use Equations (6.9) and (6.10), replacing $\mathbf{p}'_{\nabla}$ with $\mathbf{p}'_{\nabla^2}$ to obtain the Gauss-Newton system formed by $\mathbf{H_p}$ and $\mathbf{b_p}$. We compute the updated pixel location:

$$\mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla^2} + (\mathbf{H_p} + \epsilon\,\mathbf{I})^{-1}\mathbf{b_p} \tag{6.13}$$

Note that in contrast to the computation of the LM-Loss (Equation (6.12)), in this case ϵ is just added to ensure invertibility and therefore ϵ is much smaller than the λ_f used above. The Gauss-Newton loss is computed with:

$$E_{\text{GN}} = \frac{1}{2}(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}})^T\mathbf{H_p}(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}) + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H_p}|) \tag{6.14}$$

Note how all our 4 loss components use a different way to sample the involved points, depicted also in Figure 6.2. With the derivation above we argue that each loss component is important to achieve optimal performance and we demonstrate this in the results section. Note that the Gauss-Newton systems computed for the GD-Loss and the GN-Loss are very relevant for the application of direct image alignment. In fact the full Gauss-Newton system containing all points (Equation (6.3)), can be computed from these individual Gauss-Newton systems (Equation (6.10)) by simply summing them up and multiplying them with the derivative with respect to the pose [5].

6.3.3 CorrPoseNet

In order to deal with the large baselines between the images, we propose CorrPoseNet to regress the relative pose between two images I and I', which serves as the initialization of LM optimization. As our network shall work even in cases of large baselines and strong rotations, we utilize the correlation layer proposed in [109]

which is known to boost the performance of affine image transformation and optical flow [119] estimation for large displacements, but has not been applied to pose estimation before.

Our network first computes deep features $\mathbf{f}_{\text{corr}}, \mathbf{f}'_{\text{corr}} \in \mathbb{R}^{h \times w \times c}$ from both images individually using multiple strided convolutions with ReLU activations in between. Then the correlation layer correlates each pixel from the normalized source features with each pixel from the normalized target features yielding the correlation map $\mathbf{c} \in \mathbb{R}^{h \times w \times (h \times w)}$:

$$\mathbf{c}(i, j, (i', j')) = \mathbf{f}_{\text{corr}}(i, j)^T \mathbf{f}'_{\text{corr}}(i', j') \tag{6.15}$$

The correlation map is then normalized in the channel dimension and fed into 2 convolutional layers each followed by batch norm and ReLU. Finally we regress the Euler angle $\mathbf{r}^{\text{euler}}$ and translation $\mathbf{t}$ using a fully connected layer. More details on the architecture are shown in the supplementary material.

We train CorrPoseNet from scratch with image pairs and groundtruth poses $\mathbf{r}^{\text{euler}}_{\text{gt}}, \mathbf{t}_{\text{gt}}$. We utilize an L2-loss working directly on Euler angles and translation:

$$E = \|\mathbf{t} - \mathbf{t}_{\text{gt}}\|_2 + \lambda \|\mathbf{r}^{\text{euler}} - \mathbf{r}^{\text{euler}}_{\text{gt}}\|_2, \tag{6.16}$$

where λ is the weight, which we set to 10 in practice.

As the distribution of groundtruth poses in the Oxford training data is limited we apply the following data augmentation. We first generate dense depths for all training images using a state-of-the-art dense stereo matching algorithm [120]. The resulting depths are then used to warp the images to a different pose sampled from a uniform distribution. In detail, we first warp the depth image to the random target pose, then inpaint the depth image using the OpenCV implementation of Navier Stokes, and finally warp our image to the target pose using this depth map. Note that the dense depths are only necessary for training, not for evaluation. We show an ablation study on the usage of correlation layers and the proposed data augmentation in the supplementary material.

6.4 Experiments

We evaluate our method on the relocalization tracking benchmark proposed in [5], which contains images created with the CARLA simulator [81], and scenes from the Oxford RobotCar dataset [82]. We train our method on the respective datasets from scratch. LM-Net is trained using the Adam optimizer with a learning rate of 10^{-6} and for CorrPoseNet we use a learning rate of 10^{-4}. For both networks we choose hyperparameters and epoch based on the results on the validation data. Our networks use the same hyperparameters for all experiments except where stated otherwise; the direct image alignment code is slightly adapted for Oxford RobotCar, mainly to improve performance when the ego-vehicle is standing still.

As the original relocalization tracking benchmark [5] does not include validation data on Oxford RobotCar we have manually aligned two new sequences, namely *2015-04-17-09-06-25* and *2015-05-19-14-06-38*, and extend the benchmark with these sequences as validation data.

Evaluation metrics: We evaluate the predicted translation $\mathbf{t}_{\text{est}}$ and rotation $\mathbf{R}_{\text{est}}$ against the ground-truth $\mathbf{t}_{\text{gt}}$ and $\mathbf{R}_{\text{gt}}$ according to Equations (6.17) and (6.18).

$$t_{\Delta} = \|\mathbf{t}_{\text{est}} - \mathbf{t}_{\text{gt}}\|_2 \tag{6.17}$$

$$R_{\Delta} = \arccos\left(\frac{\text{trace}(\mathbf{R}_{\text{est}}^{-1}\mathbf{R}_{\text{gt}}) - 1}{2}\right) \tag{6.18}$$

In this section, we plot the cumulative translation and rotation error until 0.5m and 0.5°, respectively. For quantitative results we compute the area under curve (AUC) of these cumulative curves in percent, which we denote as t_{AUC} for translation and R_{AUC} for rotation from now on.

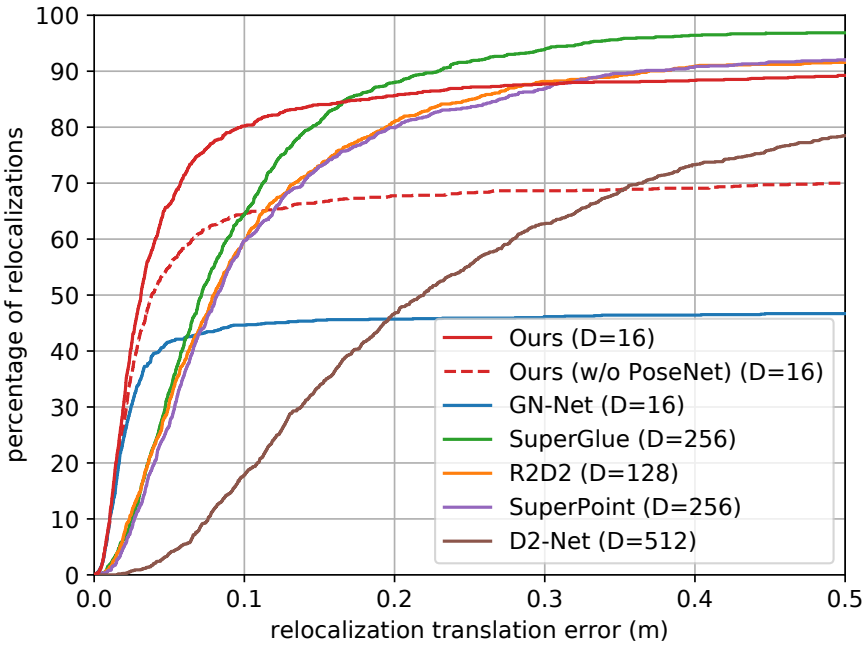
We evaluate the following direct methods:

Ours: The full LM-Reloc approach consisting of CorrPoseNet, LM-Net features and direct image alignment based on Levenberg-Marquardt. The depths used for the image alignment are estimated with the stereo version [101] of DSO [15].

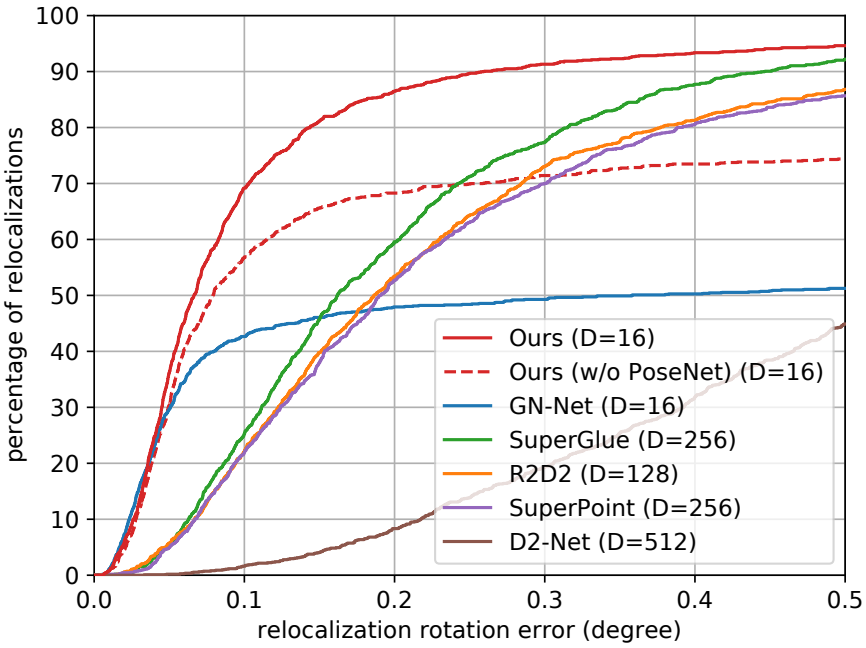
Ours (w/o CorrPoseNet): For a more fair comparison to GN-Net we use identity as initialization for the direct image alignment instead of CorrPoseNet. This enables a direct comparison between the two loss formulations.

GN-Net [5]: In this work, we have also improved the parameters of the direct image alignment pipeline based on DSO [15]. Thus we have re-evaluated GN-Net with this improved pipeline to make the comparison as fair as possible. These re-evaluated results are better than the results computed in the original GN-Net paper.

Baseline methods: Additionally, we evaluate against current state-of-the-art indirect methods, namely SuperGlue [115], R2D2 [114], SuperPoint [103], and D2-Net [102]. For these methods, we estimate the relative pose using the models provided by the authors and the OpenCV implementation of solvePnPRansac. We have tuned the parameters of RANSAC on the validation data and used 1000 iterations and a reprojection error threshold of 3 for all methods. For estimating depth values at keypoint locations we use OpenCV stereo matching. It would be possible to achieve a higher accuracy by using SfM and MVS solutions such as COLMAP [121]. However, one important disadvantage of these approaches is, that building a map is rather time consuming and computationally expensive, whereas all other approaches evaluated on the benchmark [5] are able to create the map close to real-time, enabling applications like long-term loop-closure and map-merging.



(a) Translation error.



(b) Rotation error.

Figure 6.3: Results on the CARLA relocalization tracking benchmark test data [5]. For each error threshold we show the percentage of relocalizations (cumulative error plot) for LM-Reloc (ours) and other state-of-the-art methods. Compared to the indirect methods our approach exhibits significantly better accuracy in both translation and rotation, while having a similar robustness. Compared to GN-Net, the novel loss formulation (see red dashed line), and the CorrPoseNet (see red line) both boost the robustness. D is the feature dimensionality.

Table 6.1: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the CARLA relocalization tracking benchmark test data. Powered by our novel loss formulation and the combination with CorrPoseNet, LM-Reloc achieves lower rotation and translation errors compared to the state-of-the-art.

Method	t_{AUC}	R_{AUC}
Ours	**80.65**	**77.83**
SuperGlue [115]	78.99	59.31
R2D2 [114]	73.47	54.42
SuperPoint [103]	72.76	53.38
D2-Net [102]	47.62	16.47
Ours (w/o CorrPoseNet)	63.88	61.9
GN-Net [5]	43.72	44.08

6.4.1 CARLA Relocalization Benchmark

Figure 6.3 depicts the results on the test data of the CARLA benchmark. For all methods we show the cumulative error plot for translation in meters and rotation in degree. It can be seen that our method is more accurate than the state-of-the-art while performing similarly in terms of robustness. We also show the AUC for the two Figures in Table 6.1. Compared to GN-Net it can be seen that our new loss formulation significantly improves the results, even when used without the CorrPoseNet as initialization. The figure conveys that the direct methods (Ours, GN-Net) are more accurate than the evaluated indirect methods.

6.4.2 Oxford RobotCar Relocalization Benchmark

We compare to the state-of-the-art indirect methods on the 6 test sequence pairs consisting of the sequences *2015-02-24-12-32-19* (sunny), *2015-03-17-11-08-44* (overcast), *2014-12-05-11-09-10* (rainy), and *2015-02-03-08-45-10* (snowy). In Table 6.2, we show the area under curve until 0.5 meters / 0.5 degrees for all methods. It can be seen that our method clearly outperforms the state-of-the-art in terms of rotation accuracy, while being competitive in terms of translation error. It should be noted that the ground-truth for these sequences was generated using ICP alignment of the 2D-LiDAR data accumulated for 60 meters. We have computed that the average root mean square error of the ICP alignment is 16 centimeters. Therefore, especially the ground-truth translations have limited accuracy. As can be seen from Figure 6.3, the accuracy improvements our method provides are especially visible in the range below 0.15 meters which is hard to measure on this

Table 6.2: Results on the Oxford RobotCar relocalization tracking benchmark [5]. We compare LM-Net (Ours) against other state-of-the-art methods (SuperGlue, R2D2, SuperPoint, and D2-Net). As can be seen from the results, our method almost consistently outperforms other SOTA approaches in terms of rotation AUC whilst achieving comparable results on translation AUC.

Sequence	Ours		SuperGlue [115]		R2D2 [114]		SuperPoint [103]		D2-Net [102]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	79.83	**55.48**	**81.01**	52.83	80.86	53.57	78.95	50.03	71.93	39.0
Sunny-Rainy	71.54	**43.7**	**75.58**	40.59	74.84	41.23	69.76	37.12	65.63	27.5
Sunny-Snowy	59.69	**44.06**	**63.57**	43.64	62.92	41.78	60.85	40.02	55.65	30.86
Overcast-Rainy	80.54	**63.7**	79.99	61.64	**81.29**	61.23	80.36	61.56	75.66	51.06
Overcast-Snowy	55.38	47.88	57.67	47.16	**57.68**	**48.41**	55.39	44.96	51.17	34.54
Rainy-Snowy	68.57	**41.67**	69.91	39.87	**71.79**	39.86	67.7	38.05	61.91	27.74

dataset. The rotation error of LiDAR alignment is lower than the translational one, which is why we clearly observe the improvements of our method on the rotations.

In Table 6.3, we compare LM-Net without the CorrPoseNet to GN-Net. Due to our novel loss formulation LM-Net outperforms the competitor on all sequences significantly.

6.4.3 Ablation Studies

We evaluate LM-Net on the CARLA validation data with and without the various losses (Figure 6.4). Compared to a normal contrastive loss, the given loss formulation is a large improvement. As expected, E_{GD} (green line) mainly improves the robustness, whereas E_{GN} (blue line) improves the accuracy. Only when used together (our method) we achieve large robustness and large accuracy, confirming our theoretical derivation in Section 6.3.

6.4.4 Qualitative Results

To demonstrate the accuracy of our approach in practice, we show qualitative results on the Oxford RobotCar dataset. We track the snowy test sequence *2015-02-03-08-45-10* using Stereo DSO [101] and at the same time perform relocalization against the sunny reference map *2015-02-24-12-32-19*. Relocalization between the current keyframe and the closest map image is performed using LM-Net. Initially, we give the algorithm the first corresponding map image (which would in practice be provided by an image retrieval approach such as NetVLAD [80]). Afterwards we find the closest map image for each keyframe using the previous solution for the transformation between the map and the current SLAM world T_{w_m}. We visualize the current point cloud (blue) and the point cloud from the map (grey) overlayed using the smoothed T_{w_m} (Figure 6.5). The point clouds will align only if the relocalization is accurate. As can be seen in Figure 6.5, the lane markings, poles, and buildings between the reference and query map align well, hence qualitatively showing the high relocalization accuracy of our method. We recommend watching the video at `https://vision.in.tum.de/lm-reloc`. In Figure 6.6 we show example images from the benchmark.

6.5 Conclusion

We have presented LM-Reloc as a novel approach for direct visual localization. In order to estimate the relative 6DoF pose between two images from different conditions, our approach performs direct image alignment on the trained features from LM-Net without relying on feature matching or RANSAC. In particular, with the loss function designed seamlessly for the Levenberg-Marquart algorithm,

Table 6.3: This table shows the results on the Oxford RobotCar relocalization tracking benchmark test data against GN-Net. Thanks to our LM-based loss formulation we consistently outperform GN-Net on all sequences.

Sequence	Ours (w/o CorrPoseNet)		GN-Net [5]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	**79.61**	**55.45**	73.53	49.31
Sunny-Rainy	**70.46**	**42.86**	64.58	37.27
Sunny-Snowy	**59.7**	**44.17**	55.27	41.36
Overcast-Rainy	**79.67**	**63.08**	75.72	60.13
Overcast-Snowy	**54.94**	**47.19**	51.34	42.91
Rainy-Snowy	**66.23**	**39.93**	62.63	36.2

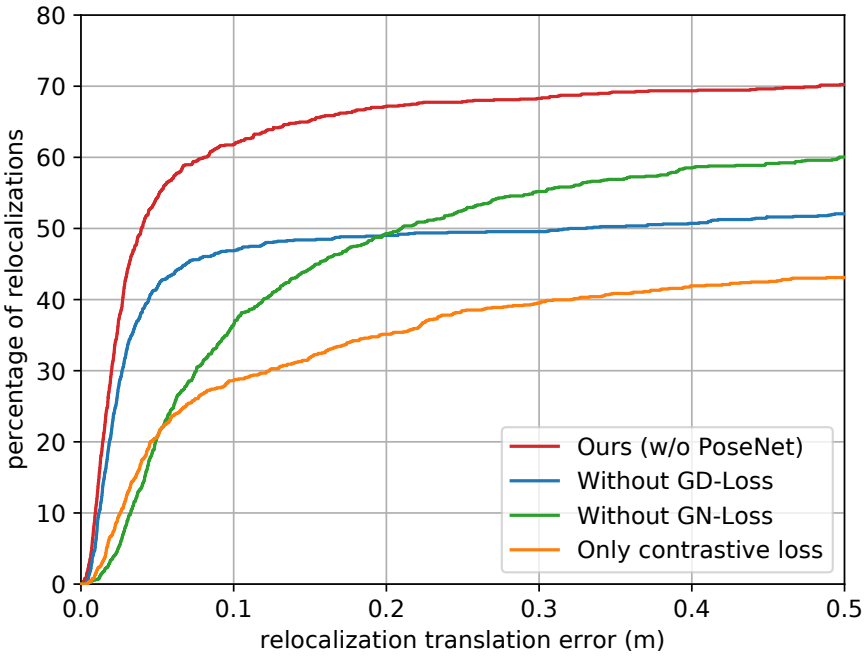


Figure 6.4: This plot shows our ablation study for removing different loss parts on the CARLA relocalization tracking benchmark. Without the GD-loss the achieved robustness is reduced, whereas removing the GN-loss leads to decreased accuracy. Using our full loss formulation yields a large improvement.

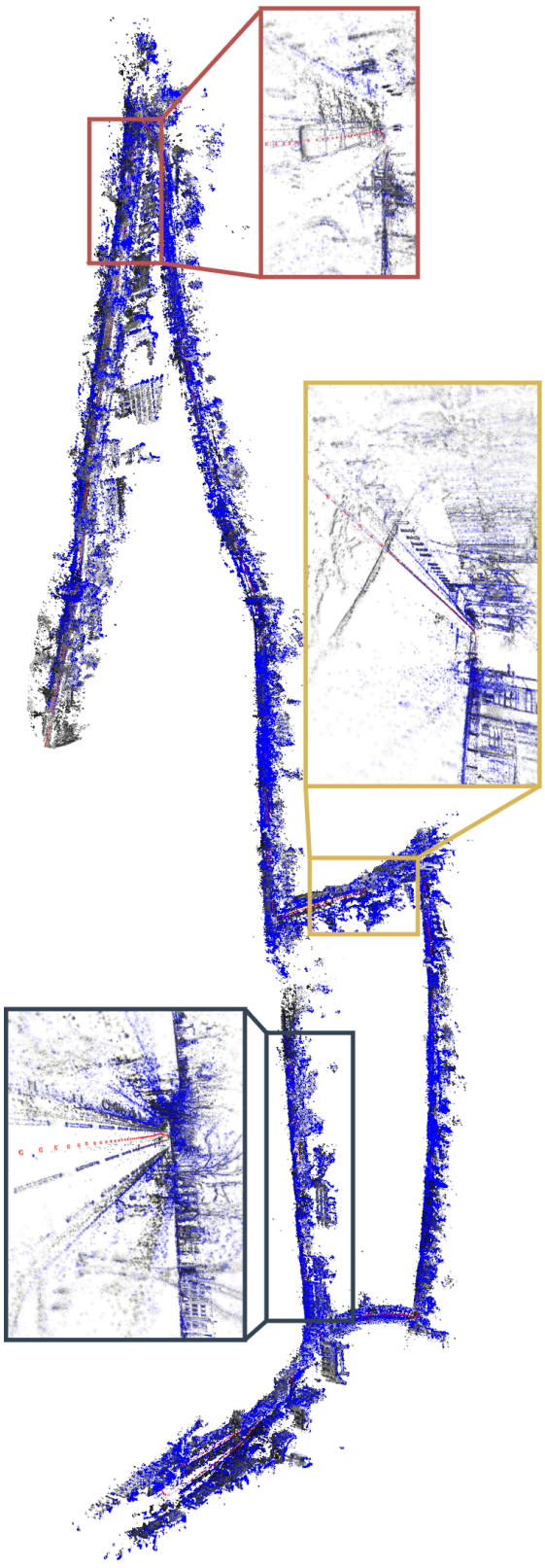


Figure 6.5: This figure shows a point cloud from a sunny reference map (grey points) overlayed with the point cloud from a relocalized snowy sequence (blue points). The well aligned point clouds demonstrate the high relocalization accuracy of LM-Reloc.

Figure 6.6: Example image pairs from the relocalization tracking benchmark which have been successfully relocalized by LM-Reloc (with an accuracy of better than 10 cm). Top row: Oxford sunny against snowy condition, middle row: Oxford sunny against rainy condition, bottom row: CARLA benchmark.

LM-Net provides deep feature maps that coin the characteristics of direct image alignment and are also invariant to changes in lighting and appearance of the scene. The experiments on the CARLA and Oxford RobotCar relocalization tracking benchmark exhibit the state-of-the-art performance of our approach. In addition, the ablation studies also show the effectiveness of the different components of LM-Reloc.

Part III

Non-Examination-Relevant Own Publications

Chapter 7

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization

Authors	Lukas von Stumberg[1]	stumberg@in.tum.de
	Vladyslav Usenko[1]	usenko@in.tum.de
	Daniel Cremers[1]	cremers@in.tum.de

[1]Technical University of Munich, Munich, Germany

Publication L. von Stumberg, V. Usenko, and D. Cremers, **Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization**, in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2510–2517. DOI: 10.1109/ICRA.2018.8462905. arXiv: 1804.05625

Contribution		
	Problem definition	*contributed*
	Literature survey	*contributed*
	Algorithm development	*significantly contributed*
	Method implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

This chapter is not examination-relevant, as this publication is based on the Master's thesis [122] of Lukas von Stumberg.

Abstract We present VI-DSO, a novel approach for visual-inertial odometry, which jointly estimates camera poses and sparse scene geometry by minimizing photometric and IMU measurement errors in a combined energy functional. The visual part of the system performs a bundle-adjustment like optimization on a sparse set of points, but unlike key-point based systems it directly minimizes a photometric error. This makes it possible for the system to track not only corners, but any pixels with large enough intensity gradients. IMU information is accumulated between several frames using measurement preintegration, and is inserted into the optimization as an additional constraint between keyframes. We explicitly include scale and gravity direction into our model and jointly optimize them together with other variables such as poses. As the scale is often not immediately observable using IMU data this allows us to initialize our visual-inertial system with an arbitrary scale instead of having to delay the initialization until everything is observable. We perform partial marginalization of old variables so that updates can be computed in a reasonable time. In order to keep the system consistent we propose a novel strategy which we call "dynamic marginalization". This technique allows us to use partial marginalization even in cases where the initial scale estimate is far from the optimum. We evaluate our method on the challenging EuRoC dataset, showing that VI-DSO outperforms the state of the art.

7.1 Introduction

Motion estimation and 3D reconstruction are crucial tasks for robots. In general, many different sensors can be used for these tasks: laser rangefinders, RGB-D cameras [46], GPS and others. Since cameras are cheap, lightweight and small passive sensors they have drawn a large attention of the community. Some examples of practical applications include robot navigation [123] and (semi)-autonomous driving [124]. However, current visual odometry methods suffer from a lack of robustness when confronted with low textured areas or fast maneuvers. To eliminate these effects a combination with another passive sensor - an inertial measurement unit (IMU) can be used. It provides accurate short-term motion constraints and, unlike vision, is not prone to outliers.

In this paper we propose a tightly coupled direct approach to visual-inertial odometry. It is based on Direct Sparse Odometry (DSO) [15] and uses a bundle-adjustment like photometric error function that simultaneously optimizes 3D geometry and camera poses in a combined energy functional. We complement the error function with IMU measurements. This is particularly beneficial for direct methods, since the error function is highly non-convex and a good initialization is important. A key drawback of monocular visual odometry is that it is not possible

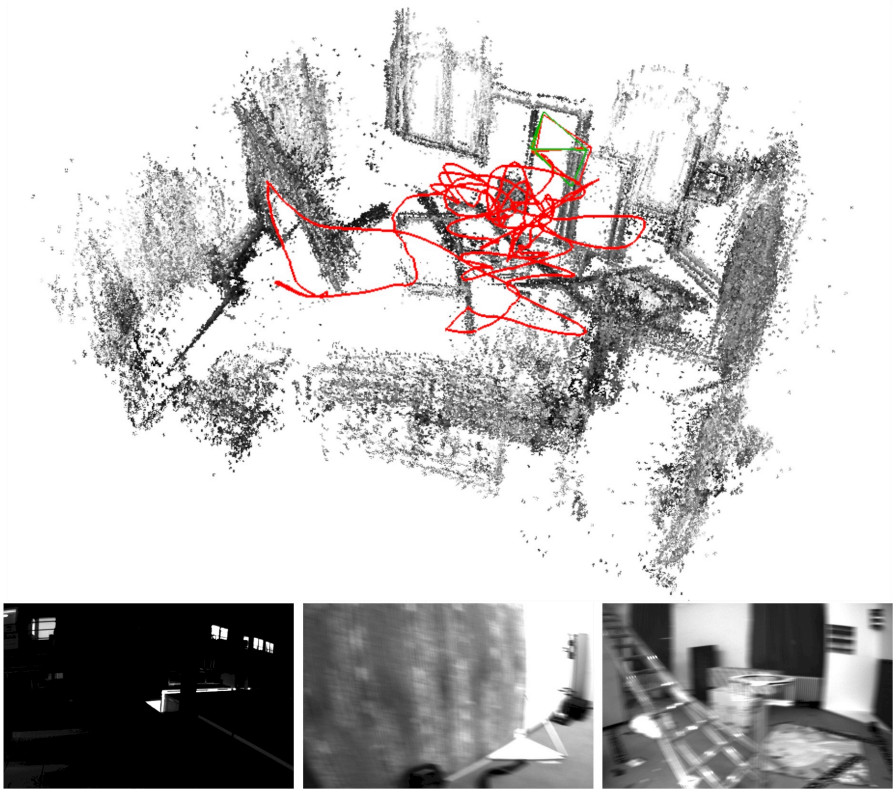


Figure 7.1: Bottom: Example images from the EuRoC-dataset: Low illumination, strong motion blur and little texture impose significant challenges for odometry estimation. Still our method is able to process all sequences with a rmse of less then 0.23m. Top: Reconstruction, estimated pose (red camera) and groundtruth pose (green camera) at the end of V1_03_difficult.

to obtain the metric scale of the environment. Adding an IMU enables us to observe the scale. Yet, depending on the performed motions this can take infinitely long, making the initialization a challenging task. Rather than relying on a separate IMU initialization we include the scale as a variable into the model of our system and jointly optimize it together with the other parameters.

Quantitative evaluation on the EuRoC dataset [70] demonstrates that we can reliably determine camera motion and sparse 3D structure (in metric units) from a visual-inertial system on a rapidly moving micro aerial vehicle (MAV) despite challenging illumination conditions (Fig. 7.1).

In summary, our contributions are:

- a direct sparse visual-inertial odometry system.
- a novel initialization strategy where scale and gravity direction are included into the model and jointly optimized after initialization.

- we introduce "dynamic marginalization" as a technique to adaptively employ marginalization strategies even in cases where certain variables undergo drastic changes.
- an extensive evaluation on the challenging EuRoC dataset showing that both, the overall system and the initialization strategy outperform the state of the art.

7.2 Related work

Motion estimation using cameras and IMUs has been a popular research topic for many years. In this section we will give a summary of visual, and visual-inertial odometry methods. We will also discuss approaches to the initialization of monocular visual-inertial odometry, where the initial orientation, velocity and scale are not known in advance.

The term visual odometry was introduced in the work of Nister et al. [14], who proposed to use frame-to-frame matching of the sparse set of points to estimate the motion of the cameras. Most of the early approaches were based on matching features detected in the images, in particular MonoSLAM [29], a real-time capable EKF-based method. Another prominent example is PTAM [25], which combines a bundle-adjustment backend for mapping with real-time capable tracking of the camera relative to the constructed map. Recently, a feature-based system capable of large-scale real-time SLAM was presented by Mur-Artal et al. [24].

Unlike feature-based methods, direct methods use un-processed intensities in the image to estimate the motion of the camera. The first real-time capable direct approach for stereo cameras was presented in [125]. Several methods for motion estimation for RGB-D cameras were developed by Kerl et al. [46]. More recently, direct approaches were also applied to monocular cameras, in a dense [30], semi-dense [32], and sparse fashion [17] [15].

Due to the complementary nature of the IMU sensors, there were many attempts to combine them with vision. They provide good short-term motion prediction and make roll and pitch angles observable. At first, vision systems were used just as a provider of 6D pose measurements which were then inserted in the combined optimization. This, so-called *loosely coupled* approach, was presented in [126] and [127]. It is generally easier to implement, since the vision algorithm requires no modifications. On the other hand, *tightly coupled* approaches jointly optimize motion parameters in a combined energy function. They are able to capture more correlations in the multisensory data stream leading to more precision and robustness. Several prominent examples are filtering based approaches [128] [16] and energy-minimization based approaches [40] [72] [41] [37].

Another issue relevant for the practical use of monocular visual-inertial odometry is initialization. Right after the start, the system has no prior information about the

initial pose, velocities and depth values of observed points in the image. Since the energy functional that is being minimized is highly non-convex, a bad initialization might result in divergence of the system. The problem is even more complicated, since some types of motion do not allow to uniquely determine all these values. A closed form solution for initialization, together with analysis of the exceptional cases was presented in [42], and extended to consider IMU biases in [43].

7.3 Direct Sparse Visual-Inertial Odometry

The following approach is based on iterative minimization of photometric and inertial errors in a non-linear optimization framework. To make the problem computationally feasible the optimization is performed on a window of recent frames while all older frames get marginalized out. Our approach is based on [15] and can be viewed as a direct formulation of [40]. In contrast to [41], we jointly determine poses and 3D geometry from a single optimization function. This results in better precision especially on hard sequences. Compared to [72] we perform a full bundle-adjustment like optimization instead of including structure-less vision error terms.

The proposed approach estimates poses and depths by minimizing the energy function

$$E_{\text{total}} = \lambda \cdot E_{\text{photo}} + E_{\text{inertial}} \tag{7.1}$$

which consists of the photometric error E_{photo} (section 7.3.2) and an inertial error term E_{inertial} (section 7.3.3).

The system contains two main parts running in parallel:

- The coarse tracking is executed for every frame and uses direct image alignment combined with an inertial error term to estimate the pose of the most recent frame.
- When a new keyframe is created we perform a visual-inertial bundle adjustment like optimization that estimates the geometry and poses of all active keyframes.

In contrast to [37] we do not wait for a fixed amount of time before initializing the visual-inertial system but instead we jointly optimize all parameters including the scale. This yields a higher robustness as inertial measurements are used right from the beginning.

7.3.1 Notation

Throughout the paper we will use the following notation: bold upper case letters $\mathbf{H}$ represent matrices, bold lower case $\mathbf{x}$ vectors and light lower case λ represent

scalars. Transformations between coordinate frames are denoted as $\mathbf{T}_{i\_j} \in \mathbf{SE}(3)$ where point in coordinate frame i can be transformed to the coordinate frame j using the following equation $\mathbf{p}_i = \mathbf{T}_{i\_j}\mathbf{p}_j$. We denote Lie algebra elements as $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$, where $\boldsymbol{\xi} \in \mathbb{R}^6$, and use them to apply small increments to the 6D pose $\boldsymbol{\xi}'_{i\_j} = \boldsymbol{\xi}_{i\_j} \boxplus \boldsymbol{\xi} := \log\left(e^{\hat{\boldsymbol{\xi}}_{i\_j}} \cdot e^{\hat{\boldsymbol{\xi}}}\right)^{\vee}$.

We define the *world* as a fixed inertial coordinate frame with gravity acting in negative Z axis. We also assume that the transformation from camera to IMU frame $T_{\text{imu\_cam}}$ is fixed and calibrated in advance. Factor graphs are expressed as a set G of factors and we use $G_1 \cup G_2$ to denote a factor graph containing all factors that are either in G_1 or in G_2.

7.3.2 Photometric Error

The photometric error of a point $p \in \Omega_i$ in reference frame i observed in another frame j is defined as follows:

$$E_{\mathbf{p}j} = \sum_{\mathbf{p}\in\mathcal{N}_{\mathbf{p}}} \omega_{\mathbf{p}} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_{\gamma}, \tag{7.2}$$

where $\mathcal{N}_{\mathbf{p}}$ is a small set of pixels around the point $\mathbf{p}$, I_i and I_j are images of respective frames, t_i, t_j are the exposure times, a_i, b_i, a_j, b_j are the coefficients to correct for affine illumination changes, γ is the Huber norm, ω_p is a gradient-dependent weighting and $\mathbf{p}'$ is the point projected into I_j.

With that we can formulate the photometric error as

$$E_{\text{photo}} = \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in\text{obs}(\mathbf{p})} E_{\mathbf{p}j}, \tag{7.3}$$

where $\mathcal{F}$ is a set of keyframes that we are optimizing, $\mathcal{P}_i$ is a sparse set of points in keyframe i, and $obs(\mathbf{p})$ is a set of observations of the same point in other keyframes.

7.3.3 Inertial Error

In order to construct the error term that depends on rotational velocities measured by the gyroscope and linear acceleration measured by the accelerometer we use the nonlinear dynamic model defined in [41, eq. (6), (7), (8)].

As IMU data is obtained with a much higher frequency than images we follow the preintegration approach proposed in [73] and improved in [74] and [72]. This allows us to add a single IMU factor describing the pose between two camera frames. For two states $\mathbf{s}_i$ and $\mathbf{s}_j$ (based on the state definition in Equation (7.9)), and IMU-measurements $\boldsymbol{a}_{i,j}$ and $\boldsymbol{\omega}_{i,j}$ between the two images we obtain a prediction $\widehat{\mathbf{s}}_j$ as well as an associated covariance matrix $\widehat{\Sigma}_{s,j}$. The corresponding error function is

$$E_{\text{inertial}}(\mathbf{s}_i, \mathbf{s}_j) := (\mathbf{s}_j \boxminus \widehat{\mathbf{s}}_j)^T \, \widehat{\boldsymbol{\Sigma}}_{s,j}^{-1} \, (\mathbf{s}_j \boxminus \widehat{\mathbf{s}}_j) \tag{7.4}$$

where the operator $\boxminus$ applies $\boldsymbol{\xi}_j \boxplus \left(\widehat{\boldsymbol{\xi}_j}\right)^{-1}$ for poses and a normal subtraction for other components.

7.3.4 IMU Initialization and the problem of observability

In contrast to a purely monocular system the usage of inertial data enables us to observe metric scale and gravity direction. This also implies that those values have to be properly initialized, otherwise optimization might diverge. Initialization of the monocular visual-inertial system is a well studied problem with an excellent summary provided in [42]. [42, Tables I and II] show that for certain motions immediate initialization is not possible, for example when moving with zero acceleration and constant non-zero velocity. To demonstrate that it is a real-world problem and not just a theoretical case we note that the state-of-the-art visual-inertial SLAM system [37] uses the first 15 seconds of camera motion for the initialization on the EuRoC dataset to make sure that all values are observable.

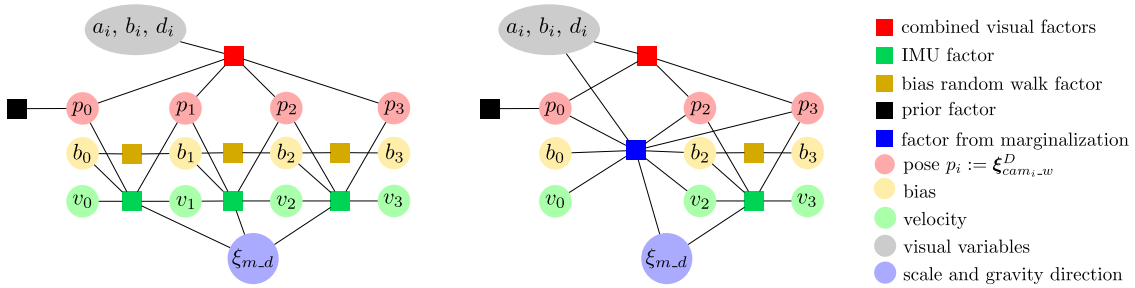
Therefore we propose a novel strategy for handling this issue. We explicitly include scale (and gravity direction) as a parameter in our visual-inertial system and jointly optimize them together with the other values such as poses and geometry. This means that we can initialize with an arbitrary scale instead of waiting until it is observable. We initialize the various parameters as follows.

- We use the same visual initializer as [15] which computes a rough pose estimate between two frames as well as approximate depths for several points. They are normalized so that the average depth is 1.
- The initial gravity direction is computed by averaging up to 40 accelerometer measurements, yielding a sufficiently good estimate even in cases of high acceleration.
- We initialize the velocity and IMU-biases with zero and the scale with 1.0.

All these parameters are then jointly optimized during a bundle adjustment like optimization.

7.3.5 SIM(3)-based Representation of the World

In order to be able to start tracking and mapping with a preliminary scale and gravity direction we need to include them into our model. Therefore in addition to the metric coordinate frame we define the DSO coordinate frame to be a scaled and rotated version of it. The transformation from the DSO frame to the metric frame is defined as $\mathbf{T}_{m\_d} \in \{\mathbf{T} \in \mathbf{SIM}(3) \mid \text{translation}(\mathbf{T}) = 0\}$, together with the corresponding $\boldsymbol{\xi}_{m\_d} = \log(\mathbf{T}_{m\_d}) \in \mathfrak{sim}(3)$. We add a superscript D or M to all poses denoting in which coordinate frame they are expressed. In the optimization



(a) Factor graph for the visual-inertial optimization.

(b) Factor graph after keyframe 1 was marginalized.

Figure 7.2: Factor graphs for the visual-inertial joint optimization before and after the marginalization of a keyframe.

the photometric error is always evaluated in the DSO frame, making it independent of the scale and gravity direction, whereas the inertial error has to use the metric frame.

7.3.6 Scale-aware Visual-inertial Optimization

We optimize the poses, IMU-biases and velocities of a fixed number of keyframes. Fig. 7.2a shows a factor graph of the problem. Note that there are in fact many separate visual factors connecting two keyframes each, which we have combined to one big factor connecting all the keyframes in this visualization. Each IMU-factor connects two subsequent keyframes using the preintegration scheme described in section 7.3.3. As the error of the preintegration increases with the time between the keyframes we ensure that the time between two consecutive keyframes is not bigger than 0.5 seconds which is similar to what [37] have done. Note that in contrast to their method however we allow the marginalization procedure described in section 7.3.6 to violate this constraint which ensures that long-term relationships between keyframes can be properly observed.

An important property of our algorithm is that the optimized poses are not represented in the metric frame but in the DSO frame. This means that they do not depend on the scale of the environment.

Nonlinear Optimization

We perform nonlinear optimization using the Gauss-Newton algorithm. For each active keyframe we define a state vector

$$\mathbf{s}_i := [\left(\boldsymbol{\xi}^D_{cam_i\_w}\right)^T, \mathbf{v}_i^T, \mathbf{b}_i^T, a_i, b_i, d_i^1, d_i^2, ..., d_i^m]^T \tag{7.5}$$

where $\mathbf{v}_i \in \mathbb{R}^3$ is the velocity, $\mathbf{b}_i \in \mathbb{R}^6$ is the current IMU bias, a_i and b_i are the affine illumination parameters used in equation (7.2) and d_i^j are the inverse depths

of the points hosted in this keyframe.

The full state vector is then defined as

$$\mathbf{s} = [\mathbf{c}^T, \boldsymbol{\xi}_{m\_d}^T, \mathbf{s}_1^T, \mathbf{s}_2^T, ..., \mathbf{s}_n^T]^T \tag{7.6}$$

where $\mathbf{c}$ contains the geometric camera parameters and $\boldsymbol{\xi}_{m\_d}$ denotes the translation-free transformation between the DSO frame and the metric frame as defined in section 7.3.5. We define the operator $\mathbf{s} \boxplus \mathbf{s}'$ to work on state vectors by applying the concatenation operation $\boldsymbol{\xi} \boxplus \boldsymbol{\xi}'$ for Lie algebra components and a plain addition for other components.

Using the stacked residual vector $\mathbf{r}$ we define

$$\mathbf{J} = \left.\frac{d\mathbf{r}\left(\mathbf{s} \boxplus \boldsymbol{\epsilon}\right)}{d\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}, \mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{7.7}$$

where $\mathbf{W}$ is a diagonal weight matrix. Then the update that we compute is $\boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b}$.

Note that the visual energy term E_{photo} and the inertial error term E_{imu} do not have common residuals. Therefore we can divide $\mathbf{H}$ and $\mathbf{b}$ each into two independent parts

$$\mathbf{H} = \mathbf{H}_{\text{photo}} + \mathbf{H}_{\text{imu}} \text{ and } \mathbf{b} = \mathbf{b}_{\text{photo}} + \mathbf{b}_{\text{imu}} \tag{7.8}$$

As the inertial residuals compare the current relative pose to the estimate from the inertial data they need to use poses in the metric frame relative to the IMU. Therefore we define additional state vectors for the inertial residuals.

$$\mathbf{s}_i' := [\boldsymbol{\xi}_{w\_imu_i}^M, \mathbf{v}_i, \mathbf{b}_i]^T \text{ and } \mathbf{s}' = \left[\mathbf{s}_1'^T, \mathbf{s}_2'^T, ..., \mathbf{s}_n'^T\right]^T \tag{7.9}$$

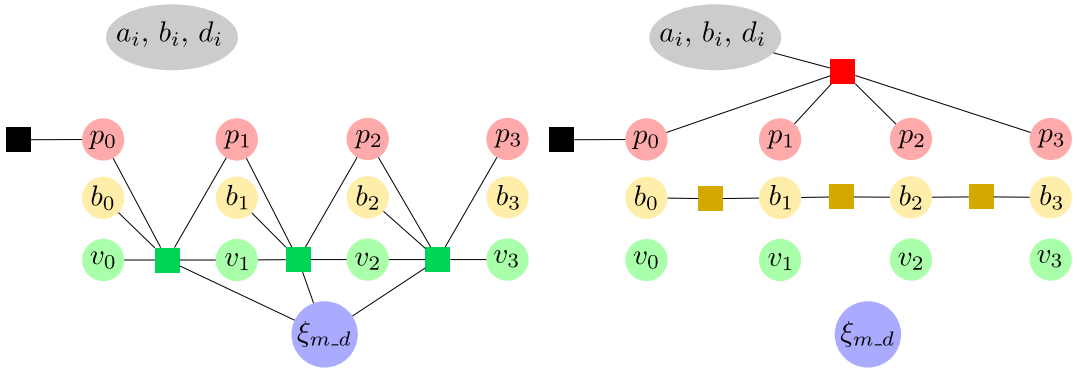
The inertial residuals lead to

$$\mathbf{H}_{\text{imu}}' = \mathbf{J}_{\text{imu}}'^T\mathbf{W}_{\text{imu}}\mathbf{J}_{\text{imu}}' \text{ and } \mathbf{b}_{\text{imu}}' = -\mathbf{J}_{\text{imu}}'^T\mathbf{W}_{\text{imu}}\mathbf{r}_{\text{imu}} \tag{7.10}$$

For the joint optimization however we need to obtain $\mathbf{H}_{\text{imu}}$ and $\mathbf{b}_{\text{imu}}$ based on the state definition in Equation (7.6). As the two definitions mainly differ in their representation of the poses we can compute $\mathbf{J}_{\text{rel}}$ such that

$$\mathbf{H}_{\text{imu}} = \mathbf{J}_{\text{rel}}^T \cdot \mathbf{H}_{\text{imu}}' \cdot \mathbf{J}_{\text{rel}} \text{ and } \mathbf{b}_{\text{imu}} = \mathbf{J}_{\text{rel}}^T \cdot \mathbf{b}_{\text{imu}}' \tag{7.11}$$

The computation of $\mathbf{J}_{\text{rel}}$ is detailed in the supplementary material. Note that we represent all transformations as elements of $\mathfrak{sim}(3)$ and fix the scale to 1 for all of them except $\boldsymbol{\xi}_{m\_d}$.



(a) G_{metric} (b) G_{visual}

Figure 7.3: Partitioning of the factor graph from Fig. 7.2a into G_{metric} and G_{visual}. G_{metric} contains all IMU-factors while G_{visual} contains the factors that do not depend on $\boldsymbol{\xi}_{m\_d}$. Note that both of them do not contain any marginalization factors.

Marginalization using the Schur-Complement

In order to compute Gauss-Newton updates in a reasonable time-frame we perform partial marginalization for older keyframes. This means that all variables corresponding to this keyframe (pose, bias, velocity and affine illumination parameters) are marginalized out using the Schur complement. Fig. 7.2b shows how marginalization changes the factor graph.

The marginalization of the visual factors is handled as in [15] by dropping residual terms that affect the sparsity of the system and by first marginalizing all points in the keyframe before marginalizing the keyframe itself.

Marginalization is performed using the Schur-complement [15, eq. (16), (17) and (18)]. As the factor resulting from marginalization requires the linearization point of all connected variables to remain fixed we apply [15, eq. (15)] to approximate the energy around further linearization points.

In order to maintain consistency of the system it is important that Jacobians are all evaluated at the same value for variables that are connected to a marginalization factor as otherwise the nullspaces get eliminated. Therefore we apply "First Estimates Jacobians". For the visual factors we follow [15] and evaluate $\mathbf{J}_{\text{photo}}$ and $\mathbf{J}_{\text{geo}}$ at the linearization point. When computing the inertial factors we fix the evaluation point of $\mathbf{J}_{\text{rel}}$ for all variables which are connected to a marginalization factor. Note that this always includes $\boldsymbol{\xi}_{m\_d}$.

Dynamic Marginalization for Delayed Scale Convergence

The marginalization procedure described in subsection 7.3.6 has two purposes: reduce the computation complexity of the optimization by removing old states and maintain the information about the previous states of the system. This procedure

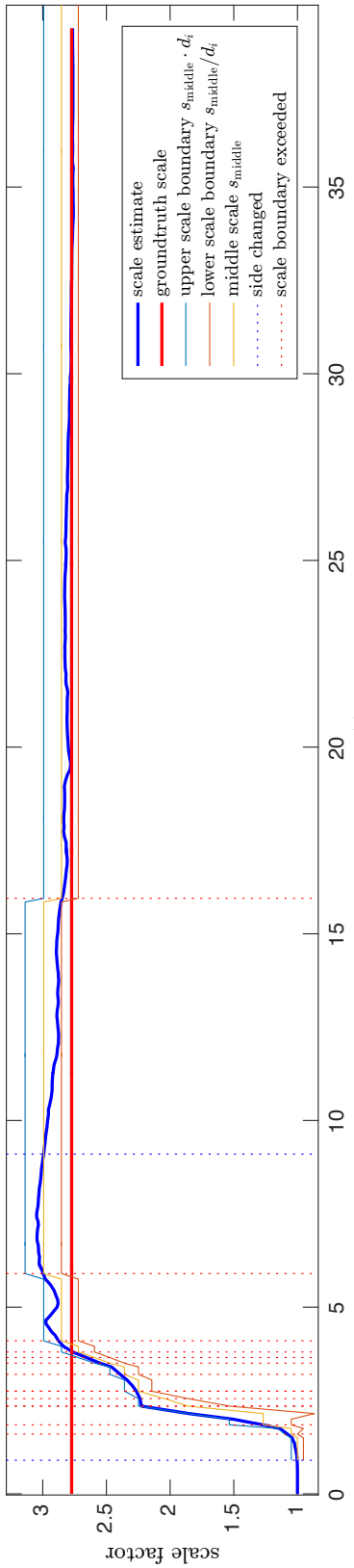


Figure 7.4: The scale estimation running on the V1_03_difficult sequence from the EuRoC dataset. We show the current scale estimate (bold blue), the groundtruth scale (bold red) and the current scale interval (light lines). The vertical dotted lines denote when the side changes (blue) and when the boundary of the scale interval is exceeded (red). In practice this means that M_{curr} contains the inertial factors since the last blue or red dotted line that is before the last red dotted line. For example at 16s it contains all inertial data since the blue line at 9 seconds.

fixes the linearization points of the states connected to the old states, so they should already have a good estimate. In our scenario this is the case for all variables except of scale.

The main idea of "Dynamic marginalization" is to maintain several marginalization priors at the same time and reset the one we currently use when the scale estimate moves too far from the linearization point in the marginalization prior.

In our implementation we use three marginalization priors: M_{visual}, M_{curr} and M_{half}. M_{visual} contains only scale independent information from previous states of the vision and cannot be used to infer the global scale. M_{curr} contains all information since the time we set the linearization point for the scale and M_{half} contains only the recent states that have a scale close to the current estimate.

When the scale estimate deviates too much from the linearization point of M_{curr}, the value of M_{curr} is set to M_{half} and M_{half} is set to M_{visual} with corresponding changes in the linearization points. This ensures that the optimization always has some information about the previous states with consistent scale estimates. In the remaining part of the section we provide the details of our implementation.

We define G_{metric} to contain only the visual-inertial factors (which depend on $\boldsymbol{\xi}_{m\_d}$) and G_{visual} to contain all other factors, except the marginalization priors. Then

$$G_{\text{full}} = G_{\text{metric}} \cup G_{\text{visual}} \tag{7.12}$$

Fig. 7.3 depicts the partitioning of the factor graph.

We define three different marginalization factors M_{curr}, M_{visual} and M_{half}. For the optimization we always compute updates using the graph

$$G_{ba} = G_{\text{metric}} \cup G_{\text{visual}} \cup M_{\text{curr}} \tag{7.13}$$

When keyframe i is marginalized we update M_{visual} with the factor arising from marginalizing frame i in $G_{\text{visual}} \cup M_{\text{visual}}$. This means that M_{visual} contains all marginalized visual factors and no marginalized inertial factors making it independent of the scale.

For each marginalized keyframe i we define

$$s_i := \text{scale estimate at the time, } i \text{ was marginalized} \tag{7.14}$$

We define $i \in M$ if and only if M contains an *inertial* factor that was marginalized at time i. Using this we enforce the following constraints for inertial factors.

$$\forall i \in M_{\text{curr}} : s_i \in [s_{\text{middle}}/d_i, s_{\text{middle}} \cdot d_i] \tag{7.15}$$

$$\forall i \in M_{\text{half}} : s_i \in \begin{cases} [s_{\text{middle}}, s_{\text{middle}} \cdot d_i], & \text{if } s_{\text{curr}} > s_{\text{middle}} \\ [s_{\text{middle}}/d_i, s_{\text{middle}}], & \text{otherwise} \end{cases} \tag{7.16}$$

where s_{middle} is the current middle of the allowed scale interval (initialized with s_0), d_i is the size of the scale interval at time i, and s_{curr} is the current scale estimate.

We update M_{curr} by marginalizing frame i in G_{ba} and we update M_{half} by marginalizing i in $G_{\text{metric}} \cup G_{\text{visual}} \cup M_{\text{half}}$

In order to preserve the constraints in Equations (7.15) and (7.16) we apply Algorithm 2 everytime a marginalization happens. By following these steps on the one hand we make sure that the constraints are satisfied which ensures that the scale difference in the currently used marginalization factor stays smaller than d_i^2. On the other hand the factor always contains some inertial factors so that the scale estimation works at all times. Note also that M_{curr} and M_{half} have separate First Estimate Jacobians that are employed when the respective marginalization factor is used. Fig. 7.4 shows how the system works in practice.

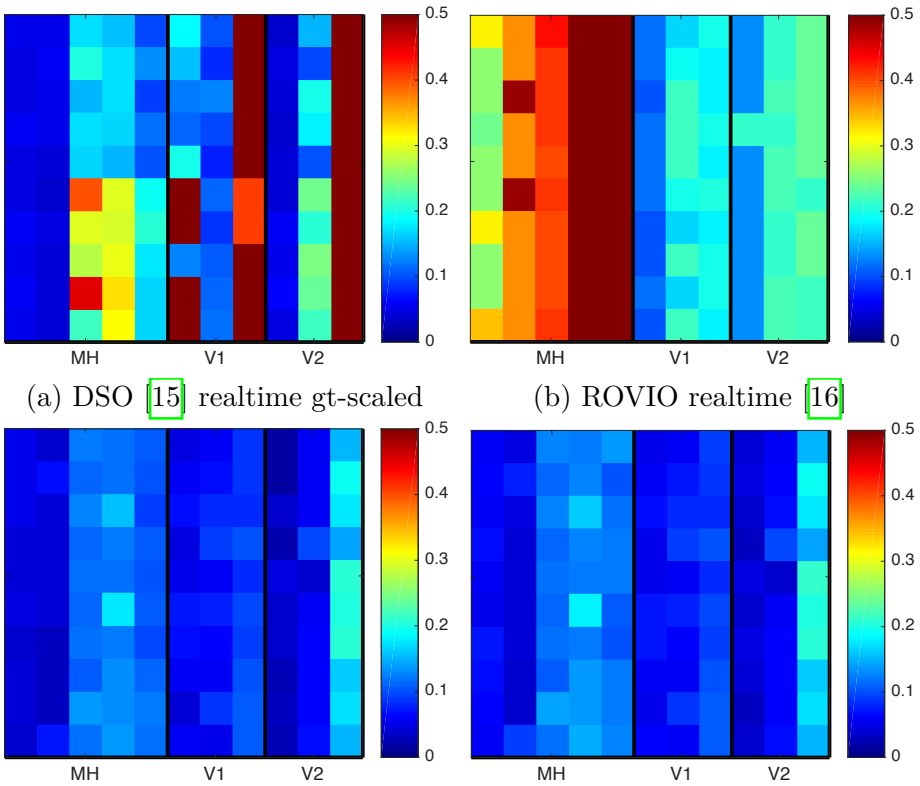
Algorithm 2 Constrain Marginalization

upper $\leftarrow s_{\text{curr}} > s_{\text{middle}}$
if upper $\neq$ lastUpper **then** ▷ Side changes.
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
end if
if $s_{\text{curr}} > s_{\text{middle}} \cdot d_i$ **then** ▷ Upper boundary exceeded.
  $M_{\text{curr}} \leftarrow M_{\text{half}}$
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
  $s_{\text{middle}} \leftarrow s_{\text{middle}} \cdot d_i$
end if
if $s_{\text{curr}} < s_{\text{middle}}/d_i$ **then** ▷ Lower boundary exceeded.
  $M_{\text{curr}} \leftarrow M_{\text{half}}$
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
  $s_{\text{middle}} \leftarrow s_{\text{middle}}/d_i$
end if
lastUpper $\leftarrow$ upper

An important part of this strategy is the choice of d_i. It should be small, in order to keep the system consistent, but not too small so that M_{curr} always contains enough inertial factors. Therefore we chose to dynamically adjust the parameter as follows. At all time steps i we calculate

$$d_i = \min \{d_{\text{min}}^j \mid j \in \mathbb{N} \setminus \{0\}, \frac{s_i}{s_{i-1}} < d_i\} \tag{7.17}$$

This ensures that it cannot happen that the M_{half} gets reset to M_{visual} at the same time that M_{curr} is exchanged with M_{half}. Therefore it prevents situations where M_{curr} contains no inertial factors at all, making the scale estimation more reliable. In our experiments we chose $d_{\text{min}} = \sqrt{1.1}$.



(a) DSO [15] realtime gt-scaled (b) ROVIO realtime [16]

(c) our method realtime gt-scaled (d) our method realtime

Figure 7.5: rmse for different methods run 10 times (lines) on each sequence (columns) of the EuRoC dataset.

7.3.7 Coarse Visual-Inertial Tracking

The coarse tracking is responsible for computing a fast pose estimate for each frame that also serves as an initialization for the joint optimization detailed in 7.3.6. We perform conventional direct image alignment between the current frame and the latest keyframe, while keeping the geometry and the scale fixed. Inertial residuals using the previously described IMU preintegration scheme are placed between subsequent frames. Everytime the joint optimization is finished for a new frame, the coarse tracking is reinitialized with the new estimates for scale, gravity direction, bias, and velocity as well as the new keyframe as a reference for the visual factors. Similar to the joint optimization we perform partial marginalization to keep the update time constrained. After estimating the variables for a new frame we marginalize out all variables except the keyframe pose and the variables of the newest frame. In contrast to the joint optimization we do not need to use dynamic marginalization because the scale is not included in the optimization.

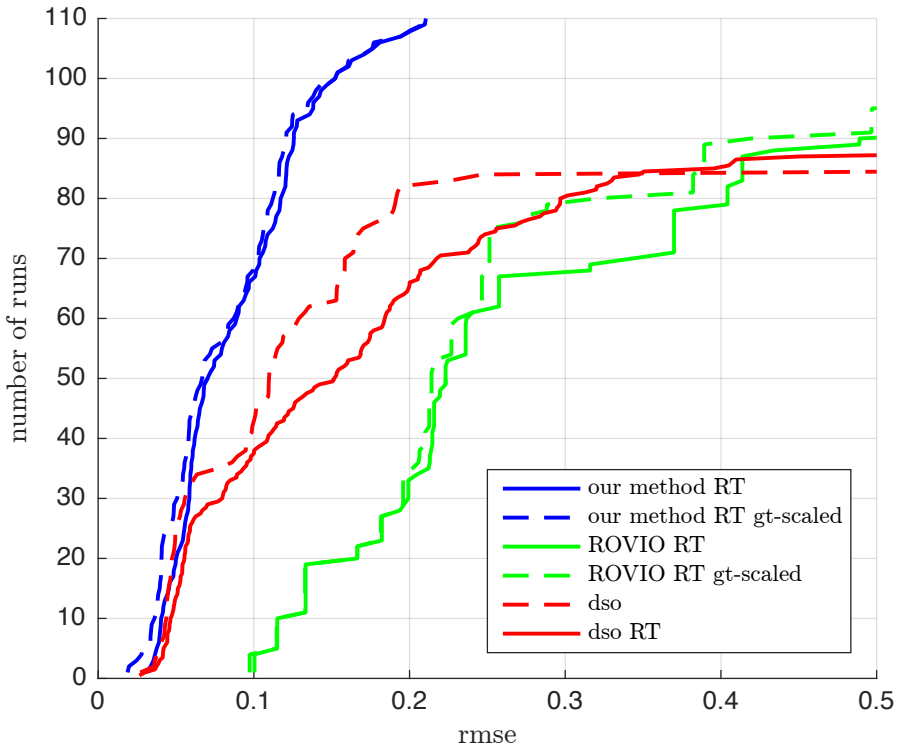


Figure 7.6: Cumulative error plot on the EuRoC-dataset (RT means realtime). This experiment demonstrates that the additional IMU not only provides a reliable scale estimate, but that it also significantly increases accuracy and robustness.

7.4 Results

We evaluate our approach on the publicly available EuRoC dataset [70]. The performance is compared to [15], [16], [37], [41], [40] and [27]. We also provide supplementary material with more evaluation and a video at vision.in.tum.de/vi-dso.

7.4.1 Robust Quantitative Evaluation

In order to obtain an accurate evaluation we run our method 10 times for each sequence of the dataset (using the left camera). We directly compare the results to visual-only DSO [15] and ROVIO [16]. As DSO cannot observe the scale we evaluate using the optimal ground truth scale in some plots (with the description "gt-scaled") to enable a fair comparison. For all other results we scale the trajectory with the final scale estimate (our method) or with 1 (other methods). For DSO we use the results published together with their paper. We use the same start and end times for each sequence to run our method and ROVIO. Note that the drone

has a high initial velocity in some sequences when using these start times making it especially challenging for our IMU initialization. Fig. 7.5 shows the root mean square error (rmse) for every run and Fig. 7.6 displays the cumulative error plot. Clearly our method significantly outperforms DSO and ROVIO. Without inertial data DSO is not able to work on all sequences especially on V1_03_difficult and V2_03_difficult and it is also not able to scale the results correctly. ROVIO on the other hand is very robust but as a filtering-based method it cannot provide sufficient accuracy.

Table 7.1 shows a comparison to several other methods. For our results we have displayed the median error for each sequence from the 10 runs plotted in Fig. 7.5c. This makes the results very meaningful. For the other methods unfortunately only one result was reported so we have to assume that they are representative as well. The results for [40] and [27] were taken from [27]. The results for [37] (as reported in their paper) differ slightly from the other methods as they show the error of the keyframe trajectory instead of the full trajectory. This is a slight advantage as keyframes are bundle-adjusted in their method which does not happen for the other frames.

In comparison to VI ORB-SLAM our method outperforms it in terms of rmse on several sequences. As ORB-SLAM is a SLAM system while ours is a pure odometry method this is a remarkable achievement especially considering the differences in the evaluation. Note that the Vicon room sequences (V*) are executed in a small room and contain a lot of loopy motions where the loop closures done by a SLAM system significantly improve the performance. Also our method is more robust as ORB-SLAM fails to track one sequence. Even considering only sequences where ORB-SLAM works our approach has a lower maximum rmse.

Compared to [40] and [27] our method obviously outperforms them. It is better than the monocular versions on every single sequence and it beats even the stereo and SLAM-versions on 9 out of 11 sequences.

In summary our method is the only one which is able to track all the sequences successfully except ROVIO.

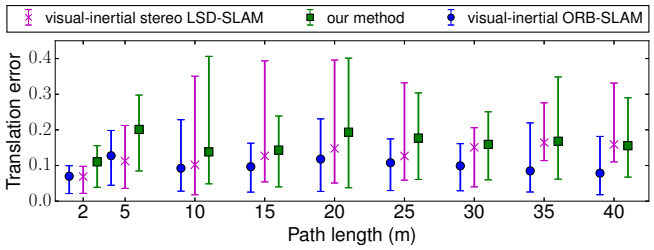
We also compare the Relative Pose Error to [37] and [41] on the V1_0*-sequences of EuRoC (Fig. 7.7). While our method cannot beat the SLAM system and the stereo method on the easy sequence we outperform [41] and are as good as [37] on the medium sequence. On the hard sequence we outperform both of the contenders even though we neither use stereo nor loop-closures.

7.4.2 Evaluation of the Initialization

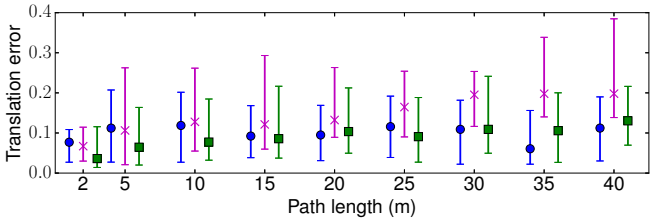
There are only few methods we can compare our initialization to. Some approaches like [42] have not been tested on real data. While [43] provides results on real data, the dataset used was featuring a downward-looking camera and an environment with a lot of features which is not comparable to the EuRoC-dataset in terms

Table 7.1: Accuracy of the estimated trajectory on the EuRoC dataset for several methods. Note that ORB-SLAM does a convincing job showing leading performance on some of the sequences. Nevertheless, since our method directly works on the sensor data (colors and IMU measurements), we observe similar precision and a better robustness – even without loop closuring. Moreover, the proposed method is the only one not to fail on any of the sequences.

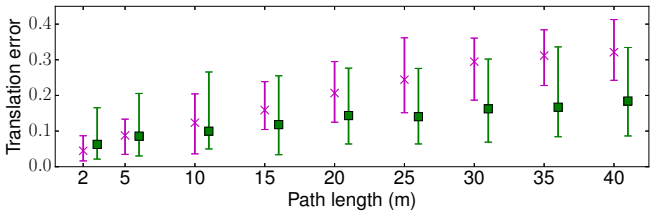
Sequence		MH1	MH2	MH3	MH4	MH5	V11	V12	V13	V21	V22	V23
VI-DSO (our method, RT) (median of 10 runs each)	RMSE	**0.062**	**0.044**	0.117	**0.132**	0.121	0.059	0.067	**0.096**	0.040	0.062	0.174
	RMSE gt-scaled	0.041	0.041	0.116	0.129	0.106	0.057	0.066	0.095	0.031	0.060	0.173
	Scale Error (%)	1.1	**0.5**	**0.4**	**0.2**	0.8	1.1	1.1	**0.8**	1.2	**0.3**	**0.4**
VI ORB-SLAM (keyframe trajectory)	RMSE	0.075	0.084	**0.087**	0.217	**0.082**	**0.027**	**0.028**	X	**0.032**	**0.041**	**0.074**
	RMSE gt-scaled	0.072	0.078	0.067	0.081	0.077	0.019	0.024	X	0.031	0.026	0.073
	Scale Error (%)	**0.5**	0.8	1.5	3.4	**0.5**	**0.9**	**0.8**	X	**0.2**	1.4	0.7
VI odometry [40], mono	RMSE	0.34	0.36	0.30	0.48	0.47	0.12	0.16	0.24	0.12	0.22	X
VI odometry [40], stereo	RMSE	0.23	0.15	0.23	0.32	0.36	0.04	0.08	0.13	0.10	0.17	X
VI SLAM [27], mono	RMSE	0.25	0.18	0.21	0.30	0.35	0.11	0.13	0.20	0.12	0.20	X
VI SLAM [27], stereo	RMSE	0.11	0.09	0.19	0.27	0.23	0.04	0.05	0.11	0.10	0.18	X



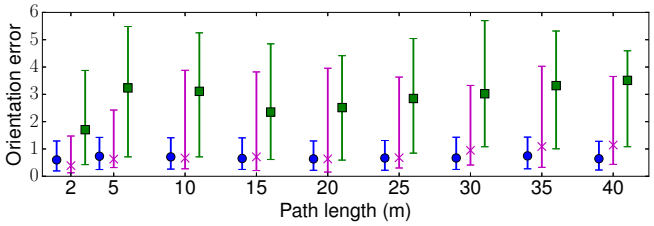
(a) Translation error V1_01_easy



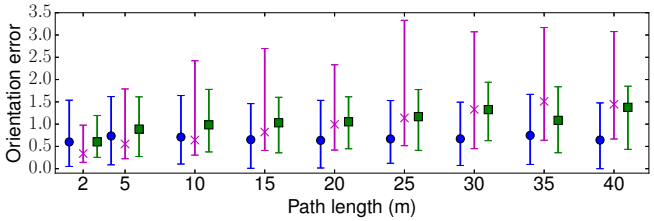
(b) Translation error V1_02_medium



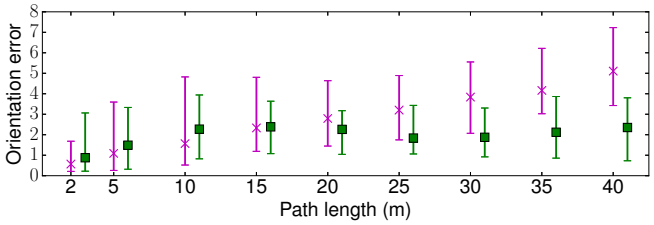
(c) Translation error V1_03_difficult



(d) Orientation error V1_01_easy



(e) Orientation error V1_02_medium



(f) Orientation error V1_03_difficult

Figure 7.7: Relative Pose Error evaluated on three sequences of the EuRoC-dataset for visual-inertial ORB-SLAM [37], visual-inertial stereo LSD-SLAM [41] and our method. Although the proposed VI-DSO does not use loop closuring (like [37]) or stereo (like [41]), VI-DSO is quite competitive in terms of accuracy and robustness. Note that [37] with loop closures is slightly more accurate on average, yet it entirely failed on V1_03_difficult.

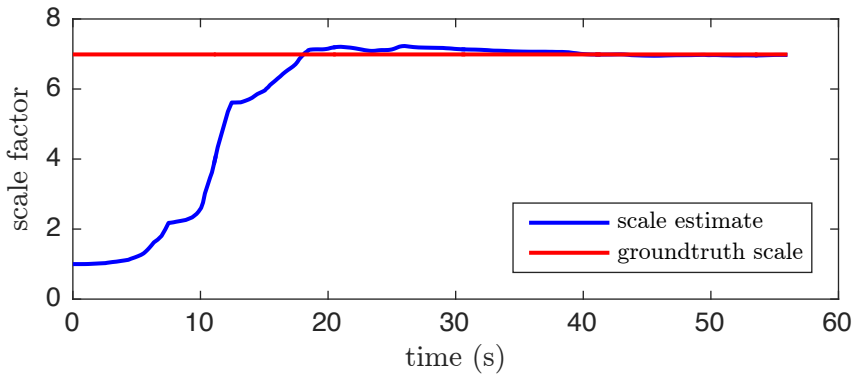


Figure 7.8: Scale estimate for MH_04_difficult (median result of 10 runs in terms of tracking accuracy). Note how the estimated scale converges to the correct value despite being initialized far from the optimum.

of difficulty. Also they do not address the problem of late observability which suggests that a proper motion is performed in the beginning of their dataset. As a filtering-based method ROVIO does not need a specific initialization procedure but it also cannot compete in terms of accuracy making it less relevant for this discussion. Visual-inertial LSD-SLAM uses stereo and therefore does not face the main problem of scale estimation. Therefore we compare our initialization procedure to visual-inertial ORB-SLAM [37] as both of the methods work on the challenging EuRoC-dataset and have to estimate the scale, gravity direction, bias, and velocity.

In comparison to [37] our estimated scale is better overall (Table 7.1). On most sequences our method provides a better scale, and our average scale error (0.7% compared to 1.0%) as well as our maximum scale error (1.2% compared to 3.4%) is lower. In addition our method is more robust as the initialization procedure of [37] fails on V1_03_difficult.

Apart from the numbers we argue that our approach is superior in terms of the general structure. While [37] have to wait for 15 seconds until the initialization is performed, our method provides an approximate scale and gravity direction almost instantly, that gets enhanced over time. Whereas in [37] the pose estimation has to work for 15 seconds without any IMU data, in our method the inertial data is used to improve the pose estimation from the beginning. This is probably one of the reasons why our method is able to process V1_03_difficult. Finally our method is better suited for robotics applications. For example an autonomous drone is not able to fly without gravity direction and scale for 15 seconds and hope that afterwards the scale was observable. In contrast our method offers both of them right from the start. The continuous rescaling is also not a big problem as an application could use the unscaled measurements for building a consistent map and for providing flight goals, whereas the scaled measurements can be used for

the controller. Fig. 7.8 shows the scale estimation for MH_04.

Overall we argue that our initialization procedure exceeds the state of the art and think that the concept of initialization with a very rough scale estimate and jointly estimating it during pose estimation will be a useful concept in the future.

7.5 Conclusion

We have presented a novel formulation of direct sparse visual-inertial odometry. We explicitely include scale and gravity direction in our model in order to deal with cases where the scale is not immediately observable. As the initial scale can be very far from the optimum we have proposed a novel technique called dynamic marginalization where we maintain multiple marginalization priors and constrain the maximum scale difference. Extensive quantitative evaluation demonstrates that the proposed visual-inertial odometry method outperforms the state of the art, both the complete system as well as the IMU initialization procedure. In particular, experiments confirm that the inertial information not only provides a reliable scale estimate, but it also drastically increases precision and robustness.

Acknowledgements We thank Jakob Engel for releasing the code of DSO and for his helpful comments on First Estimates Jacobians, and the authors of [24] for providing their numbers for the comparison in Fig. 7.7.

Chapter 8

D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry

Authors	Nan Yang[1,2]	yangn@in.tum.de
	Lukas von Stumberg[1,2]	stumberg@in.tum.de
	Rui Wang[1,2]	wangr@in.tum.de
	Daniel Cremers[1,2]	cremers@in.tum.de

[1]Technical University of Munich, Munich, Germany
[2]Artisense

Publication: N. Yang, L. von Stumberg, R. Wang, and D. Cremers, **D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry**, in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1278–1289. DOI: 10.1109/CVPR42600.2020.00136. arXiv: 2003.01060

Contribution		
	Problem definition	*contributed*
	Literature survey	*helped*
	Algorithm development	*contributed*
	Method implementation	*contributed*
	Experimental evaluation	*helped*
	Preparation of the manuscript	*helped*

This chapter is not examination-relevant, as this is not a first-author publication.

Abstract We propose D3VO as a novel framework for monocular visual odometry that exploits deep networks on three levels – deep depth, pose and uncertainty estimation. We first propose a novel self-supervised monocular depth estimation network trained on stereo videos without any external supervision. In particular, it aligns the training image pairs into similar lighting condition with predictive brightness transformation parameters. Besides, we model the photometric uncertainties of pixels on the input images, which improves the depth estimation accuracy and provides a learned weighting function for the photometric residuals in direct (feature-less) visual odometry. Evaluation results show that the proposed network outperforms state-of-the-art self-supervised depth estimation networks. D3VO tightly incorporates the predicted depth, pose and uncertainty into a direct visual odometry method to boost both the front-end tracking as well as the back-end non-linear optimization. We evaluate D3VO in terms of monocular visual odometry on both the KITTI odometry benchmark and the EuRoC MAV dataset. The results show that D3VO outperforms state-of-the-art traditional monocular VO methods by a large margin. It also achieves comparable results to state-of-the-art stereo/LiDAR odometry on KITTI and to the state-of-the-art visual-inertial odometry on EuRoC MAV, while using only a single camera.

8.1 Introduction

Deep learning has swept most areas of computer vision – not only high-level tasks like object classification, detection and segmentation [129]–[131], but also low-level ones such as optical flow estimation [132], [133] and interest point detection and description [103], [134], [135]. Yet, in the field of Simultaneously Localization And Mapping (SLAM) or Visual Odometry (VO) which estimates the relative camera poses from image sequences, traditional geometric-based approaches [15], [31], [32] still dominate the field. While monocular methods [15], [24] have the advantage of low hardware cost and less calibration effort, they cannot achieve competitive performance compared to stereo [31], [101] or visual-inertial odometry (VIO) [36], [37], [40], [7], due to the scale drift [21], [108] and low robustness. Recently, there have been many efforts to address this by leveraging deep neural networks [136]–[139]. It has been shown that with deep monocular depth estimation networks [55], [140]–[142], the performance of monocular VO is boosted, since deep networks are able to estimate depth maps with consistent metric scale by learning a-priori knowledge from a large amount of data [143].

In this way, however, deep neural networks are only used to a limited degree. Recent advances of self- and unsupervised monocular depth estimation networks [52], [141] show that the poses of the adjacent monocular frames can be predicted together

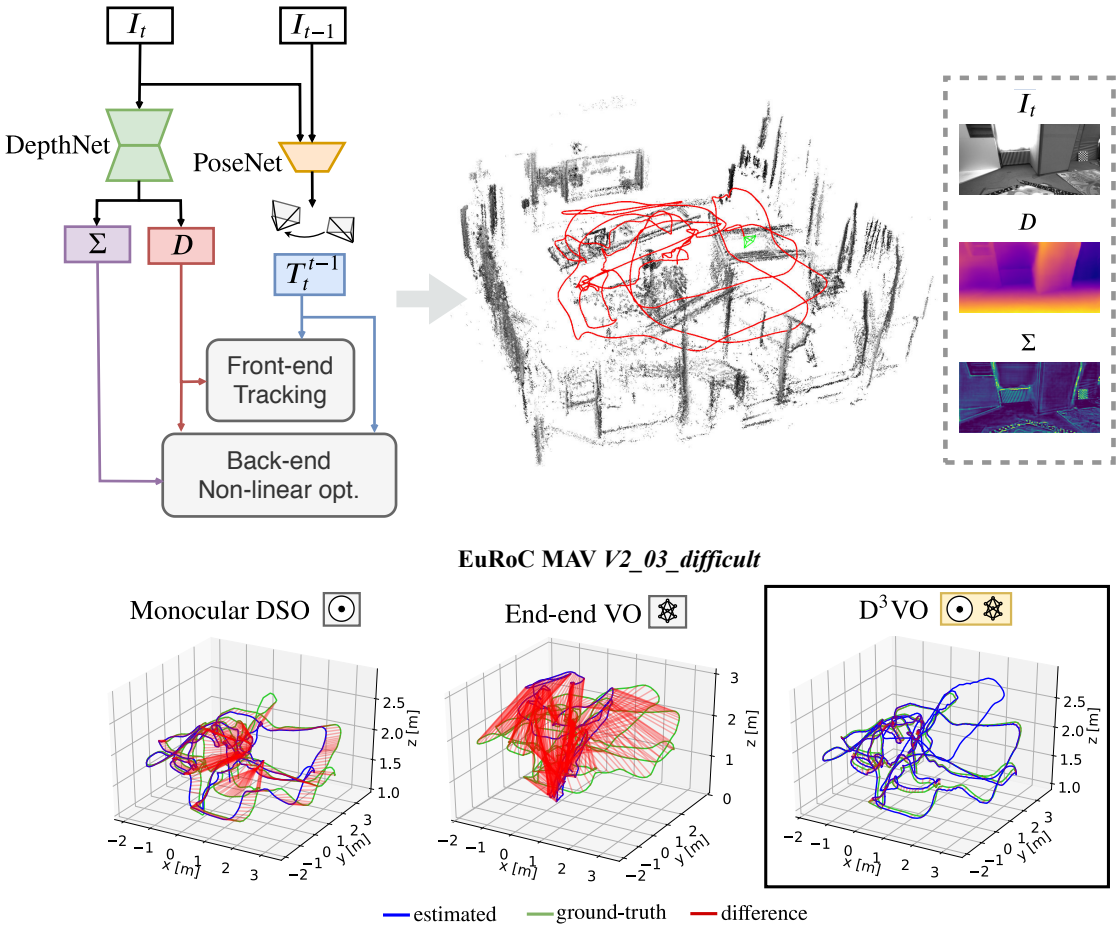


Figure 8.1: We propose D3VO – a novel monocular visual odometry (VO) framework which exploits deep neural networks on three levels: **D**eep depth (D), **D**eep pose (T_t^{t-1}) and **D**eep uncertainty (Σ) estimation. D3VO integrates the three estimations tightly into both the front-end tracking and the back-end non-linear optimization of a sparse direct odometry framework [15].

with the depth. Since the pose estimation from deep neural networks shows high robustness, one question arises: *Can the deep-predicted poses be employed to boost traditional VO*? On the other hand, since SLAM/VO is essentially a state estimation problem where uncertainty plays an important role [18], [144], [145] and meanwhile many learning based methods have started estimating uncertainties, the next question is, *how can we incorporate such uncertainty-predictions into optimization-based VO*?

In this paper, we propose D3VO as a framework for monocular direct (feature-less) visual VO that exploits self-supervised monocular depth estimation network on three levels: *deep depth*, *pose* and *uncertainty* estimation, as shown in Fig. 8.1. To this end, we first propose a purely self-supervised network trained with stereo videos. The proposed self-supervised network predicts the depth from a single

image with DepthNet and the pose between two adjacent frames with PoseNet. The two networks are bridged by minimizing the photometric error originated from both *static* stereo warping with the rectified baseline and *temporal* warping using the predicted pose. In this way, the temporal information is incorporated into the training of depth, which leads to more accurate estimation. To deal with the inconsistent illumination between the training image pairs, our network predicts the *brightness transformation parameters* which align the brightness of source and target images during training on the fly. The evaluation on the EuRoC MAV dataset shows that the proposed brightness transformation significantly improves the depth estimation accuracy. To integrate the deep depth into VO system, we firstly initialize every new 3D point with the predicted depth with a metric scale. Then we adopt the *virtual stereo term* proposed in Deep Virtual Stereo Odometry (DVSO) [55] to incorporate the predicted pose into the non-linear optimization. Unlike DVSO which uses a semi-supervised monocular depth estimation network relying on auxiliary depth extracted from state-of-the-art stereo VO system [101], our network uses only stereo videos without any external depth supervision.

Although the illumination change is explicitly modeled, it is not the only factor which may violate the brightness constancy assumption [146]. Other factors, e.g., non-Lambertian surfaces, high-frequency areas and moving objects, also corrupt it. Inspired by the recent research on aleatoric uncertainty by deep neural networks [146], [147], the proposed network estimates the photometric uncertainty as predictive variance conditioned on the input image. As a result, the errors originated from pixels which are likely to violate the brightness constancy assumption are down-weighted. The learned weights of the photometric residuals also drive us to the idea of incorporating it into direct VO – since both the self-supervised training scheme and the direct VO share a similar photometric objective, we propose to use the learned weights to replace the weighting function of the photometric residual in traditional direct VO which is empirically set [148] or only accounts for the intrinsic uncertainty of the specific algorithm itself [15], [22].

Robustness is one of the most important factors in designing VO algorithm. However, traditional monocular visual VO suffers from a lack of robustness when confronted with low textured areas or fast movement [7]. The typical solution is to introduce an inertial measurement unit (IMU). But this increases the calibration effort and, more importantly, at constant velocity, IMUs cannot deliver the metric scale in constant velocity [42]. We propose to increase the robustness of monocular VO by incorporating the estimated pose from the deep network into both the front-end tracking and the back-end non-linear optimization. For the front-end tracking, we replace the pose from the constant velocity motion model with the estimated pose from the network. Besides, the estimated pose is also used as a squared regularizer in addition to direct image alignment [149]. For the back-end non-linear optimization, we propose a pose energy term which is jointly minimized with the photometric energy term of direct VO.

We evaluate the proposed monocular depth estimation network and D3VO on both KITTI [124] and EuRoC MAV [70]. We achieve state-of-the-art performances on both monocular depth estimation and camera tracking. In particular, by incorporating deep depth, deep uncertainty and deep pose, D3VO achieves comparable results to state-of-the-art stereo/LiDAR methods on KITTI Odometry, and also comparable results to the state-of-the-art VIO methods on EuRoC MAV, while being a monocular method.

8.2 Related Work

Deep learning for monocular depth estimation. Supervised learning [142], [150], [151] shows great performance on monocular depth estimation. Eigen et al. [150], [152] propose to use multi-scale CNNs which directly regresses the pixel-wise depth map from a single input image. Laina et al. [142] propose a robust loss function to improve the estimation accuracy. Fu et al. [153] recast the monocular depth estimation network as an ordinal regression problem and achieve superior performance. More recent works start to tackle the problem in a self- and unsupervised way by learning the depth map using the photometric error [52], [53], [140], [154]–[157] and adopting differentiable interpolation [158]. Our self-supervised depth estimation network builds upon MonoDepth2 [141] and extends it by predicting the brightness transformation parameters and the photometric uncertainty.

Deep learning for uncertainty estimation. The uncertainty estimation of deep learning has recently been investigated in [147], [159] where two types of uncertainties are proposed. Klodt et al. [146] propose to leverage the concept of aleatoric uncertainty to estimate the photometric and the depth uncertainties in order to improve the depth estimation accuracy. However, when formulating the photometric uncertainty, they do not consider brightness changes across different images which in fact can be modeled explicitly. Our method predicts the photometric uncertainty conditioned on the brightness-aligned image, which can deliver better photometric uncertainty estimation. Besides, we also seek to make better use of our learned uncertainties and propose to incorporate them into traditional VO systems [15].

Deep learning for VO / SLAM. End-to-end learned deep neural networks have been explored to directly predict the relative poses between images with supervised [49], [50], [160] or unsupervised learning [51]–[53], [156]. Besides pose estimation, CodeSLAM [33] delivers dense reconstruction by jointly optimizing the learned prior of the dense geometry together with camera poses. However, in terms of pose estimation accuracy all these end-to-end methods are inferior to classical stereo or visual inertial based VO methods. Building on the success of deep monocular depth estimation, several works integrate the predicted depth/disparity map into monocular VO systems [55], [136] to improve performance and eliminate

the scale drift. CNN-SLAM [136] fuses the depth predicted by a supervised deep neural network into LSD-SLAM [32] and the depth maps are refined with Bayesian filtering, achieving superior performance in indoor environments [161], [162]. Other works [163], [164] explore the application of deep neural networks on feature based methods ,and [165] uses Generative Adversarial Networks (GANs) as an image enhancement method to improve the robustness of VO in low light. The most related work to ours is Deep Virtual Stereo Odometry (DVSO). DVSO proposes a virtual stereo term that incooperates the depth estimation from a semi-supervised network into a direct VO pipeline. In particular, DVSO outperforms other monocular VO systems by a large margin, and even achieves comparable performance to state-of-the-art stereo visual odometry systems [31], [101]. While DVSO merely leverages the depth, the proposed D3VO exploits the power of deep networks on multiple levels thereby incorporating more information into the direct VO pipeline.

8.3 Method

We first introduce a novel self-supervised neural network that predicts depth, pose and uncertainty. The network also estimates *affine brightness transformation parameters* to align the illumination of the training images in a self-supervised manner. The photometric uncertainty is predicted based on a distribution over the possible brightness values [146], [147] for each pixel. Thereafter we introduce D3VO as a direct visual odometry framework that incorporates the predicted properties into both the tracking front-end and the photometric bundle adjustment backend.

8.3.1 Self-supervised Network

The core concept of the proposed monocular depth estimation network is the self-supervised training scheme which simultaneously learns depth with DepthNet and motion with PoseNet using video sequences [52], [141]. The self-supervised training is realized by minimizing the minimum of the photometric re-projection errors between the temporal and static stereo images:

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t, I_{t' \to t}). \tag{8.1}$$

where V is the set of all pixels on I_t and t' is the index of all source frames. In our setting I_t is the left image and $I_{t'}$ contains its two adjacent temporal frames and its opposite (right) frame, i.e., $I_{t'} \in \{I_{t-1}, I_{t+1}, I_{t^s}\}$. The per-pixel minimum loss is proposed in Monodepth2 [141] in order to handle the occlusion among different source frames. To simplify notation, we use I instead of $I(\mathbf{p})$ in the remainder of this section. $I_{t' \to t}$ is the sythesized I_t by warping the temporal stereo images with the predicted depth D_t, the camera pose $\mathbf{T}_t^{t'}$, the camera intrinsics K, and

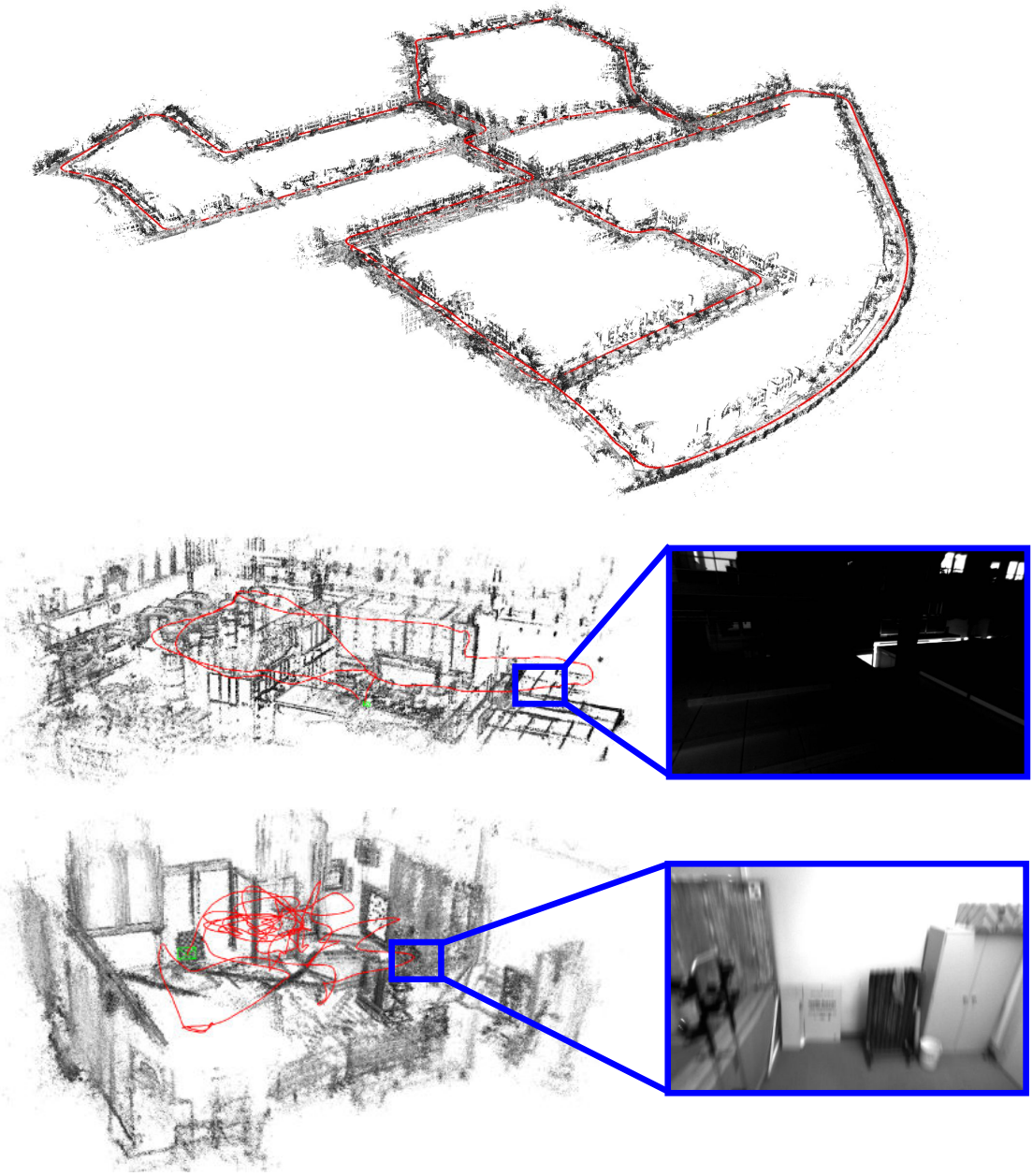


Figure 8.2: Examples of point clouds and trajectories delivered by D3VO on KITTI Odometry Seq. 00, EuRoC *MH_05_difficult* and *V1_03_difficult*. The insets on EuRoC show the scenarios with low illumination and motion blur which are among the main reasons causing failures of traditional purely vision-based VO systems.

the differentialble bilinear sampler [158]. Note that for $I_{t^s \to t}$, the transformation $\mathbf{T}_t^{t^s}$ is known and constant. DepthNet also predicts the depth map D_{t^s} of the right image I_{t^s} by feeding only the left image I_t as proposed in [140]. The training of D_{t^s} requires to synthesize $I_{t \to t^s}$ and compare with I_{t^s}. For simplicity, we will in the following only detail the loss regarding the left image.

The common practice [140] is to formulate the photometric error as

$$r(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1 - \alpha)||I_a - I_b||_1 \tag{8.2}$$

based on the brightness constancy assumption. However, it can be violated due to illumination changes and auto-exposure of the camera to which both L1 and SSIM [166] are not invariant. Therefore, we propose to explicitly model the camera exposure change with predictive *brightness transformation parameters*.

Brightness transformation parameters. The change of the image intensity due to the adjustment of camera exposure can be modeled as an affine transformation with two parameters a, b

$$I^{a,b} = aI + b. \tag{8.3}$$

Despite its simplicity, this formulation has been shown to be effective in direct VO/SLAM, e.g., [15], [101], [167], [168], which builds upon the brightness constancy assumption as well. Inspired by these works, we propose predicting the transformation parameters a, b which align the brightness condition of I_t with $I_{t'}$. We reformulate Eq. (8.1) as

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t^{a_{t'}, b_{t'}}, I_{t' \to t}) \tag{8.4}$$

with

$$I_t^{a_{t'}, b_{t'}} = a_{t \to t'} I_t + b_{t \to t'}, \tag{8.5}$$

where $a_{t \to t'}$ and $b_{t \to t'}$ are the transformation parameters aligning the illumination of I_t to $I_{t'}$. Note that both parameters can be trained in a self-supervised way without any supervisional signal. Fig. 8.3 shows the affine transformation examples from EuRoC MAV [70].

Photometric uncertainty. Only modeling affine brightness change is not enough to capture all failure cases of the brightness constancy assumption. Other cases like non-Lambertian surfaces and moving objects, are caused by the intrinsic properties of the corresponding objects which are not trivial to model analytically [146]. Since these aspects can be seen as observation noise, we leverage the concept of heteroscedastic aleatoric uncertainty of deep neural networks proposed by Kendall et al. [147]. The key idea is to predict a posterior probability distribution for each pixel parameterized with its mean as well as its variance $p(y|\tilde{y}, \sigma)$ over ground-truth labels y. For instance, by assuming the noise is Laplacian, the

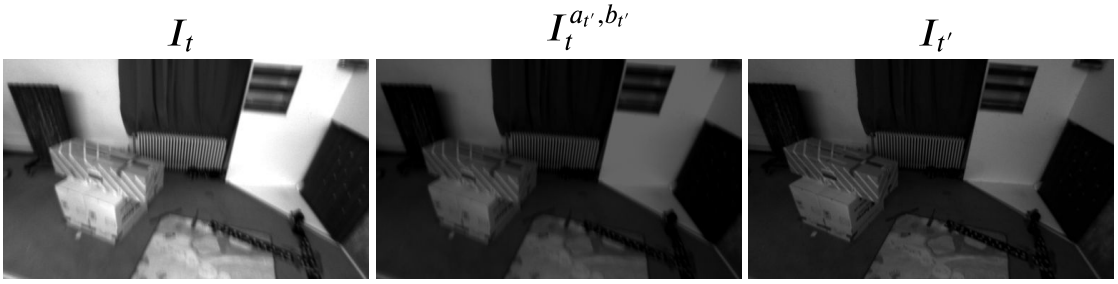


Figure 8.3: Examples of affine brightness transformation on EuRoC MAV [70]. Originally the source image ($I_{t'}$) and the target image (I_t) show different brightness. With the predicted parameters a, b, the transformed target images $I^{a',b'}$ have similar brightness as the source images, which facilitates the self-supervised training based on the brightness constancy assumption.

negative log-likelihood to be minimized is

$$-\log p(y|\tilde{y},\sigma) = \frac{|y-\tilde{y}|}{\sigma} + \log\sigma + const. \tag{8.6}$$

Note that no ground-truth label for σ is needed for training. The predictive uncertainty allows the network to adapt the weighting of the residual dependent on the data input, which improves the robustness of the model to noisy data or erroneous labels [147].

In our case where the "ground-truth" y are the pixel intensities on the target images, the network will predict higher σ for the pixel areas on I_t where the brightness constancy assumption may be violated. Similar to [146], we implement this by converting Eq. (8.4) to

$$L_{self} = \frac{1}{|V|}\sum_{\mathbf{p}\in V}\frac{\min_{t'} r(I_t^{a_{t'},b_{t'}}, I_{t'\to t})}{\Sigma_t} + \log\Sigma_t, \tag{8.7}$$

where Σ_t is the uncertainty map of I_t. Fig. 8.4 shows the qualitative results of the predicted uncertainty maps on KITTI [124] and EuRoC [70] datasets, respectively. In the next section, we will show that the learned Σ_t is useful for weighting the photometric residuals for D3VO.

The total loss function is the summation of the self-supervised losses and the regularization losses on multi-scale images:

$$L_{total} = \frac{1}{s}\sum_{s}(L_{self}^s + \lambda L_{reg}^s), \tag{8.8}$$

where $s = 4$ is the number of scales and

$$L_{reg} = L_{smooth} + \beta L_{ab} \tag{8.9}$$

with

$$L_{ab} = \sum_{t'} (a_{t'} - 1)^2 + b_{t'}^2 \tag{8.10}$$

is the regularizer of the brightness parameters and L_{smooth} is the edge-aware smoothness on D_t [140].

To summarize, the proposed DepthNet predicts D_t, D_{t^s} and Σ_t with one single input I_t. PoseNet predicts $\mathbf{T}_t^{t'}$, $a_{t\to t'}$ and $b_{t\to t'}$ with channel-wise concatenated (I_t, $I_{t'}$) as the input. Both DepthNet and PoseNet are convolutional networks following the widely used UNet-like architecture [97]. Please refer to our supplementary materials for network architecture and implementation details.

8.3.2 D3VO

In the previous section, we introduced the self-supervised depth estimation network which predicts the depth map D, the uncertainty map Σ and the relative pose $\mathbf{T}_t^{t'}$. In this section, we will describe how D3VO integrates these predictions into a windowed sparse photometric bundle adjustment formulation as proposed in [15]. Note that in the following we use $\widetilde{\cdot}$ denoting the predictions from the network as $\widetilde{D}$, $\widetilde{\Sigma}$ and $\widetilde{\mathbf{T}}_t^{t'}$ to avoid ambiguity.

Photometric energy. D3VO aims to minimize a total photometric error E_{photo} defined as

$$E_{photo} = \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in \text{obs}(\mathbf{p})} E_{\mathbf{p}j}, \tag{8.11}$$

where $\mathcal{F}$ is the set of all keyframes, $\mathcal{P}_i$ is the set of points hosted in keyframe i, obs($\mathbf{p}$) is the set of keyframes in which point $\mathbf{p}$ is observable and $E_{\mathbf{p}j}$ is the weighted photometric energy term when $\mathbf{p}$ is projected onto keyframe j:

$$E_{\mathbf{p}j} := \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} w_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{e^{a_j}}{e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma, \tag{8.12}$$

where $\mathcal{N}$ is the set of 8 neighboring pixels of $\mathbf{p}$ defined in [15], a,b are the affine brightness parameters jointly estimated by non-linear optimization as in [15] and $||\cdot||_\gamma$ is the Huber norm. In [15], the residual is down-weighted when the pixels are with high image gradient to compensate small independent geometric noise [15]. In realistic scenarios, there are more sources of noise, e.g., reflection[146], that need to be modeled in order to deliver accurate and robust motion estimation. We propose to use the learned uncertainty $\widetilde{\Sigma}$ to formulate the weighting function

$$w_\mathbf{p} = \frac{\alpha^2}{\alpha^2 + \left\| \widetilde{\Sigma}(\mathbf{p}) \right\|_2^2}, \tag{8.13}$$

which may not only depend on local image gradient, but also on higher level noise pattern. As shown in Fig. 8.4, the proposed network is able to predict high uncertainty on the areas of reflectance, e.g., the windows of the vehicles, the moving object like the cyclist and the object boundaries where depth discontinuity occurs.

The projected point position of $\mathbf{p}'$ is given by $\mathbf{p}' = \Pi(\mathbf{T}_i^j \Pi^{-1}(\mathbf{p}, d_\mathbf{p}))$, where $d_\mathbf{p}$ is the depth of the point $\mathbf{p}$ in the coordinate system of keyframe i and $\Pi(\cdot)$ is the projection function with the known camera intrinsics. Instead of randomly initializing $d_\mathbf{p}$ as in traditional monocular direct methods [15], [32], we initialize the point with $d_\mathbf{p} = \widetilde{D}_i[\mathbf{p}]$ which provides the metric scale. Inspired by [55], we introduce a *virtual stereo term* $E_\mathbf{p}^\dagger$ to Eq. (8.11)

$$E_{photo} = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \left(\lambda E_\mathbf{p}^\dagger + \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j} \right) \tag{8.14}$$

with

$$E_\mathbf{p}^\dagger = w_\mathbf{p} \left\| I_i^\dagger[\mathbf{p}^\dagger] - I_i[\mathbf{p}] \right\|_\gamma, \tag{8.15}$$

$$I_i^\dagger[\mathbf{p}^\dagger] = I_i[\Pi(\mathbf{T_s}^{-1} \Pi^{-1}(\mathbf{p}^\dagger, D_{i^s}[\mathbf{p}^\dagger]))] \tag{8.16}$$

with $\mathbf{T_s}$ the transformation matrix from the left to the right image used for training DepthNet and

$$\mathbf{p}^\dagger = \Pi(\mathbf{T_s} \Pi^{-1}(\mathbf{p}, d_\mathbf{p})). \tag{8.17}$$

The virtual stereo term optimizes the estimated depth $d_\mathbf{p}$ from VO to be consistent with the depth predicted by the proposed deep network [55].

Pose energy. Unlike traditional direct VO approaches [17], [18] which initialize the front-end tracking for each new frame with a constant velocity motion model, we leverage the predicted poses between consecutive frames to build a non-linear factor graph [169], [170]. Specifically, we create a new factor graph whenever the newest keyframe, which is also the reference frame for the front-end tracking, is updated. Every new frame is tracked with respect to the reference keyframe with direct image alignment [149]. Additionally, the predicted relative pose from the deep network is used as a factor between the current frame and the last frame. After the optimization is finished, we marginalize the last frame and the factor graph will be used for the front-end tracking of the following frame. Please refer to our supp. materials for the visualization of the factor graph.

The pose estimated from the tracking front-end is then used to initialize the photometric bundle adjustment backend. We further introduce a prior for the relative keyframe pose $\mathbf{T}_{i-1}^i$ using the predicted pose $\widetilde{\mathbf{T}}_{i-1}^i$. Note that $\widetilde{\mathbf{T}}_{i-1}^i$ is calculated by concatenating all the predicted frame-to-frame poses between keyframe $i-1$ and i. Let

$$E_{pose} = \sum_{i \in \mathcal{F} - \{0\}} \text{Log}(\widetilde{\mathbf{T}}_{i-1}^i \mathbf{T}_i^{i-1})^\top \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\xi}}_{i-1}^i}^{-1} \text{Log}(\widetilde{\mathbf{T}}_{i-1}^i \mathbf{T}_i^{i-1}), \tag{8.18}$$

where Log: $\mathrm{SE}(3) \to \mathbb{R}^6$ maps from the transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$ in the Lie group SE(3) to its corresponding twist coordinate $\boldsymbol{\xi} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$. The diagonal inverse covariance matrix $\boldsymbol{\Sigma}^{-1}_{\widetilde{\boldsymbol{\xi}}^i_{i-1}}$ is obtained by propagating the covariance matrix between each consecutive frame pairs that is modeled as a constant diagonal matrix.

The total energy function is defined as

$$E_{total} = E_{photo} + wE_{pose}. \tag{8.19}$$

Including the pose prior term E_{pose} in Eq. 8.19 can be considered as an analogy to integrating the pre-integrated IMU pose prior into the system with a Gaussian noise model. E_{total} is minimized using the Gauss-Newton method. To summarize, we boost the direct VO method by introducing the predicted poses as initializations to both the tracking front-end and the optimization backend, as well as adding them as a regularizer to the energy function of the photometric bundle adjustment.

8.4 Experiments

We evaluate the proposed self-supervised monocular depth estimation network as well as D3VO on both the KITTI [124] and the EuRoC MAV [70] datasets.

8.4.1 Monocular Depth Estimation

KITTI. We train and evalutate the proposed self-supervised depth estimation network on the split of Eigen at el. [150]. The network is trained on stereo sequences with the pre-processing proposed by Zhou et al. [52], which gives us 39,810 training quadruplets, each of which contains 3 (left) temporal images and 1 (right) stereo image, and 4,424 for validation. The upper part of Table 8.1 shows the comparison with Monodepth2 [141] which is the state-of-the-art method trained with stereo and monocular setting, and also the ablation study of the proposed brightness transformation prediction (*ab*) and the photometric uncertainty estimation (*uncer*). The results demonstrate that the proposed depth estimation network outperforms Monodepth2 on all metrics. The ablation studies unveil that the significant improvement over Monodepth2 comes largely with *uncer*, possibly because in KITTI there are many objects with non-Lambertian surfaces like windows and also objects that move independently such as cars and leaves which violate the brightness constancy assumption. The lower part of the table shows the comparison to the state-of-the-art *semi*-supervised methods and the results show that our method can achieve competitive performance without using any depth supervision.

In Figure 8.4 we show some qualitative results obtained from the Eigen test set [150]. From left to right, the original image, the depth maps and the uncertainty

Table 8.1: Depth evaluation results on the KITTI Eigen split [150]. M: self-supervised monocular supervision; S: self-supervised stereo supervision; D: ground-truth depth supervison; D*: sparse auxiliary depth supervision. The upper part shows the comparison with the SOTA self-supervised network Monodepth2 [141] under the same setting and the ablation study of the brightness transformation parameters (*ab*) and the photometric uncertainty (*uncer*). The lower part shows the comparison with the SOTA *semi*-supervised methods using stereo as well as depth supervision. Our method outperforms Monodepth2 on all metrics and can also deliver comparable performance to the SOTA semi-supervised method DVSO [55] that additionally uses the depth from Stereo DSO [101] as sparse supervision signal.

		RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Approach	Train	lower is better				higher is better		
MonoDepth2[140]	MS	4.750	0.196	0.106	0.818	0.874	*0.957*	0.979
Ours, *uncer*	MS	*4.532*	*0.190*	*0.101*	*0.772*	*0.884*	0.956	*0.978*
Ours, *ab*	MS	4.650	0.193	0.105	0.791	0.878	*0.957*	0.979
Ours, *full*	MS	**4.485**	**0.185**	**0.099**	**0.763**	**0.885**	**0.958**	**0.979**
Kuznietsov et al. [143]	DS	4.621	0.189	0.113	*0.741*	0.862	**0.960**	**0.986**
DVSO [55]	D*S	**4.442**	*0.187*	**0.097**	**0.734**	**0.888**	0.958	*0.980*
Ours	MS	*4.485*	**0.185**	*0.099*	0.763	*0.885*	0.958	0.979

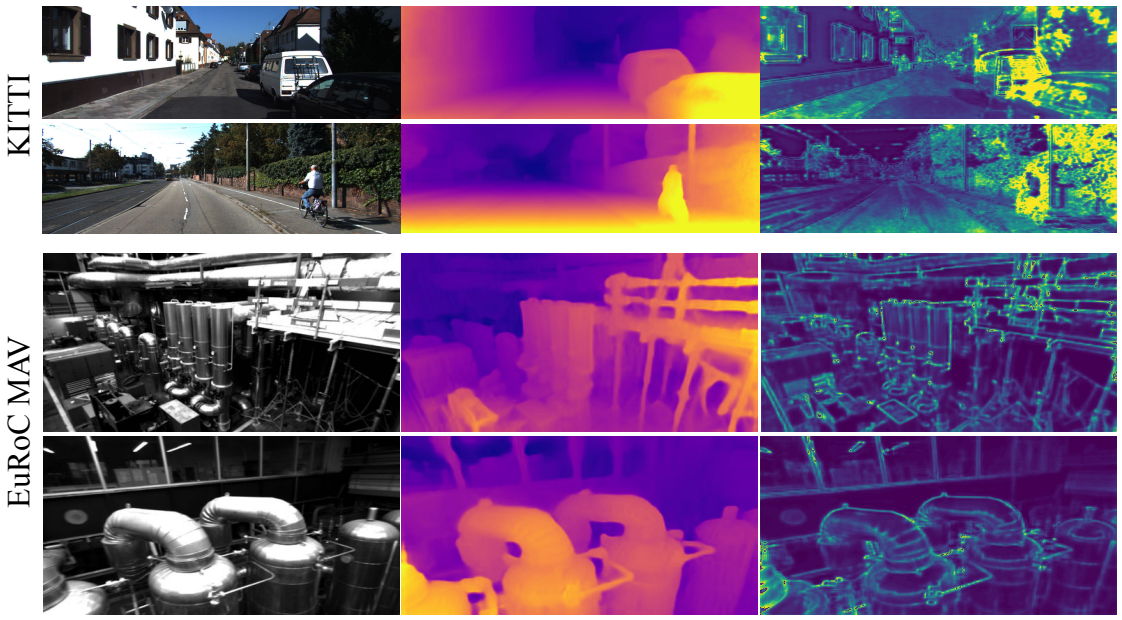


Figure 8.4: Qualitative results from KITTI and EuRoC MAV. The original image, the predicted depth maps and the uncertainty maps are shown from the left to the right, respectively. In particular, the network is able to predict high uncertainty on object boundaries, moving objects, highly reflecting and high frequency areas.

maps are shown respectively. For more qualitative results and the generalization capability on the Cityscapses dataset [171], please refer to our supp. materials.

EuRoC MAV. The EuRoC MAV Dataset [70] is a dataset containing 11 sequences categorized as *easy*, *medium* and *difficult* according to the illumination and camera motion. This dataset is very challenging due to the strong motion and significant illumination changes both between stereo and temporal images. We therefore consider it as a nice test bench for validating the effectiveness of our predictive brightness transformation parameters for depth prediction. Inspired by Gordon et al. [157] who recently generated ground truth depth maps for the sequence *V2_01* by projecting the provided Vicon 3D scans and filtering out occluded points, we also use this sequence for depth evaluations[1]. Our first experiment is set up to be consistent as in [157], for which we train models with the monocular setting on all *MH* sequences and test on *V2_01* and show the results in Table 8.3.

In the second experiment, we use 5 sequences *MH_01*, *MH_02*, *MH_04*, *V1_01* and *V1_02* as the training set to check the performance of our method in a relatively loosened setting. We remove the static frames for training and this results in 12,691 images of which 11,422 images are used for training and 1269 images are used for validation. We train our model with different ablations, as

[1]We thank the authors of [157] to provide the processing code.

Table 8.2: Evaluation results of *V2_01* in EuRoC MAV [70]. The performance of monocular depth estimation is boosted largely by the proposed predictive brightness transformation parameters.

	RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$
Monodepth2	0.370	0.148	0.102	0.065	0.890
Ours, *ab*	*0.339*	*0.130*	*0.086*	*0.054*	*0.929*
Ours, *uncer*	0.368	0.144	0.100	0.065	0.892
Ours, *full*	**0.337**	**0.128**	**0.082**	**0.051**	**0.931**

Table 8.3: Evaluation results of *V2_01* in EuRoC MAV [70] with the model trained with all *MH* sequences.

	RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$
[157]	0.971	0.396	0.332	0.389	0.420
Ours	**0.943**	**0.391**	**0.330**	**0.375**	**0.438**

well as Monodepth2 [141] as the baseline. The results in Table 8.2 show that all our variations outperform the baseline and, in contrast to the case in KITTI, the proposed *ab* improves the results on this dataset significantly. Please refer to the supp. materials for more experiments on *ab*. In fact, it is worth noting that the results in Table 8.3 (trained on one scene *MH* and tested on another scene *V*) are worse than the ones in Table 8.2 (trained on both *MH* and *V*), which implies that it is still a challenge to improve the generalization capability of monocular depth estimation among very different scenarios.

8.4.2 Monocular Visual Odometry

We evaluate the VO performance of D3VO on both KITTI Odometry and EuRoC MAV with the network trained on the splits described in the previous section.

KITTI Odometry. The KITTI Odometry Benchmark contains 11 (0-10) sequences with provided ground-truth poses. As summarized in [55], sequences 00, 03, 04, 05, 07 are in the training set of the Eigen split that the proposed network uses, so we consider the rest of the sequences as the testing set for evaluating the pose estimation of D3VO. We use the relative translational (t_{rel}) error proposed in [124] as the main metric for evaluation. Table 8.4 shows the comparison with other state-of-the-art *mono* (M) as well as *stereo* (S) VO methods on the rest of the sequences. We refer to [55] for the results of the compared methods. Traditional monocular methods show high errors in the large-scale outdoor scene like the sequences in KITTI due to the scale drift. D3VO achieves the best performance on average, despite being a monocular methods as well. The table also contains the ablation study on the integration of deep depth (*Dd*), pose (*Dp*) and uncertainty

Table 8.4: Results on our test split of KITTI Odometry. The results of the SOTA monocular (M) methods are shown as baselines. The comparison with the SOTA stereo (S) methods shows that D3VO achieves better average performance than other methods, while being a monocular VO. We also show the ablation study for the integration of deep depth(*Dd*), pose(*Dp*) as well as uncertainty(*Du*).

		01	02	06	08	09	10	mean
M	DSO [15]	9.17	114	42.2	177	28.1	24.0	65.8
	ORB [24]	108	10.3	14.6	11.5	9.30	2.57	37.0
S	S. LSD [167]	2.13	1.09	1.28	1.24	1.22	0.75	1.29
	ORB2 [31]	1.38	0.81	0.82	1.07	0.82	*0.58*	0.91
	S. DSO [101]	1.43	**0.78**	*0.67*	**0.98**	0.98	**0.49**	0.89
	Dd	1.16	0.84	0.71	1.01	0.82	0.73	0.88
	Dd+Dp	1.15	0.84	0.70	1.03	*0.80*	0.72	0.87
	Dd+Du	*1.10*	0.81	*0.69*	1.03	**0.78**	*0.62*	*0.84*
	D3VO	**1.07**	*0.80*	**0.67**	*1.00*	*0.78*	*0.62*	**0.82**

Table 8.5: Comparison to other hybrid methods as well as end-to-end methods on Seq.09 and 10 of KITTI Odometry.

		Seq. 09	Seq. 10
End-to-end	UnDeepVO [51]	7.01	10.63
	SfMLearner [52]	17.84	37.91
	Zhan et al. [53]	11.92	12.45
	Struct2Depth [172]	10.2	28.9
	Bian et al. [173]	11.2	10.1
	SGANVO [174]	*4.95*	**5.89**
	Gordon et al. [157]	**2.7**	*6.8*
Hybrid	CNN-SVO [137]	10.69	4.84
	Yin et al. [139]	4.14	1.70
	Zhan et al. [138]	2.61	2.29
	DVSO [55]	*0.83*	*0.74*
	D3VO	**0.78**	**0.62**

(*Du*). It can be noticed that, consistent with the results in Table 8.1, the predicted uncertainty helps a lot on KITTI. We also submit the results on the testing sequences (11-20) to the KITTI Odometry evaluation server (link). At the time of submission, D3VO outperforms DVSO and achieves the best monocular VO performance and comparable to other state-of-the-art LiDAR and stereo methods.

We further compare D3VO with state-of-the-art end-to-end deep learning methods and other recent hybrid methods and show the results in Table 8.5. Note that here we only show the results on Seq.09 and 10, since most of the end-to-end methods only provide the results on these two sequences. We refer to [55], [138], [157] for the results for the compared methods. D3VO achieves better performance than all the end-to-end methods by a notable margin. In general, hybrid methods which combine deep learning with traditional methods deliver better results than end-to-end methods.

EuRoC MAV. As introduced in Sec. 4.1, EuRoC MAV is very challenging for purely vision-based VO due to the strong motion and significant illumination changes. VIO methods [34], [36], [40], [7] dominate this benchmark by integrating IMU measurements to get a pose or motion prior and meanwhile estimating the absolute scale. We compare D3VO with other state-of-the-art monocular VIO (M+I) as well as stereo VIO (S+I) methods on sequences *MH_03_medium*, *MH_05_difficult*, *V1_03_difficult*, *V2_02_medium* and *V2_03_difficult*. All the other sequences are used for training. We refer to [76] for the results of the M+I methods. The results of DSO and ORB-SLAM are shown as baselines. We also show the results from the proposed PoseNet *(End-end VO)*. For the evaluation metric, we use the root mean square (RMS) of the absolute trajectory error (ATE) after aligning the estimates with ground truth. The results in Table 8.6 show that with the proposed framework integrating depth, pose and uncertainty from the proposed deep neural network, D3VO shows high accuracy as well as robustness and is able to deliver comparable results to other state-of-the-art VIO methods with only a single camera. We also show the ablation study for the integration of predicted depth (*Dd*), pose (*Dp*) and uncertainty (*Du*) and the integration of pose prediction improves the performance significantly on *V1_03_difficult* and *V2_03_difficult* where violent camera motion occurs.

Figure 8.5 shows the qualitative comparison of trajectories obtained from DSO [15], ORB-SLAM [24], visual inertial DSO [7], the end-to-end predicted poses from our network and D3VO on the *MH_03* and *V1_03* sequences. All the 5 methods can deliver fairly good results on *MH_05_difficult*. On *V1_03_difficult* where the motions are stronger and there are many brightness inconsistencies between temporal and stereo images, D3VO can still deliver comparable results to VI-DSO, while using only a single camera.

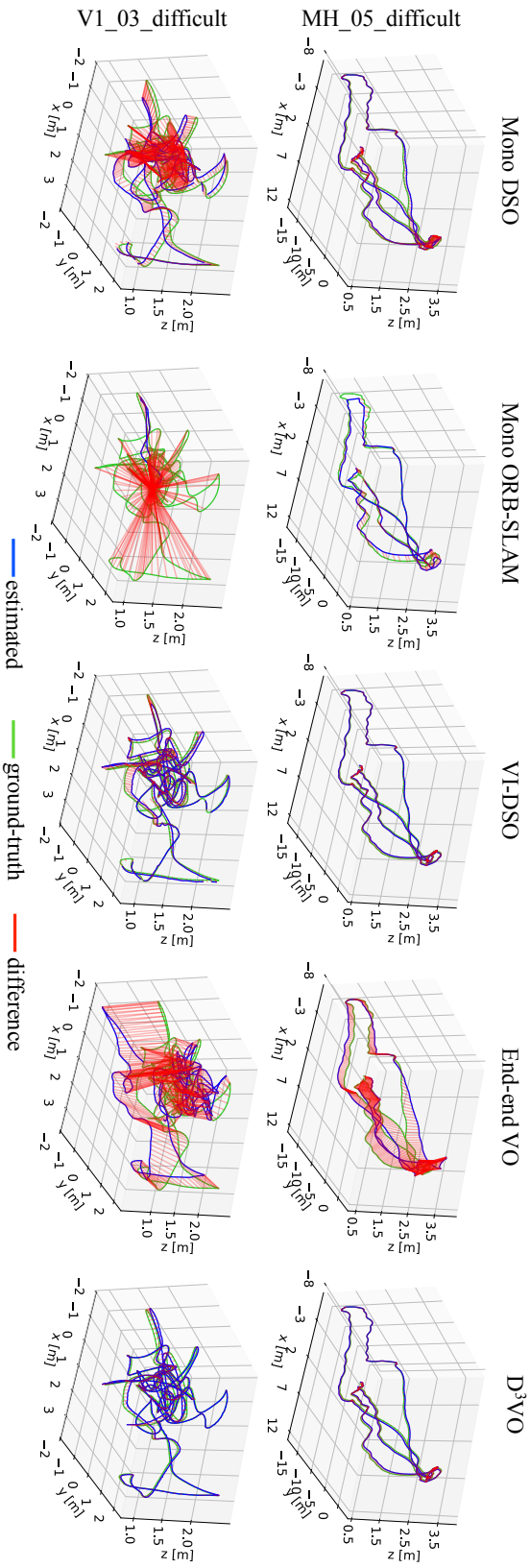


Figure 8.5: Qualitative comparison of the trajectories on *MH_05_difficult* and *V1_03_difficult* from EuRoC MAV.

Table 8.6: Evaluation results on EuRoC MAV [70]. We show the results of DSO and ORB-SLAM as baselines and compare D3VO with other SOTA monocular VIO (M+I) and stereo VIO (S+I) methods. Note that for stereo methods, *V2_03_difficult* is excluded due to many missing images from one of the cameras [34]. Despite being a monocular method, D3VO shows comparable results to SOTA monocular/stereo VIO. The best results among the monocular methods are shown as **black bold** and the best among the stereo methods are shown as **blue bold**. The ablation study shows that *Dd+Dp* delivers large improvement on *V1_03_difficult* and *V2_03_difficult* where the camera motions are very strong.

		M03	M05	V103	V202	V203	mean
M	DSO [15]	0.18	0.11	1.42	0.12	0.56	0.48
	ORB [24]	**0.08**	0.16	1.48	1.72	0.17	0.72
M+I	VINS [175]	0.13	0.35	0.13	0.08	0.21	0.18
	OKVIS [40]	0.24	0.47	0.24	0.16	0.29	0.28
	ROVIO [16]	0.25	0.52	0.14	0.14	0.14	0.24
	MSCKF [39]	0.23	0.48	0.24	0.16	*0.13*	0.25
	SVO [176]	0.12	0.16	X	X	X	0.14+X
	VI-ORB [37]	*0.09*	**0.08**	X	**0.04**	**0.07**	0.07+X
	VI-DSO [7]	0.12	0.12	**0.10**	0.06	*0.17*	*0.11*
	End-end VO	1.80	0.88	1.00	1.24	0.78	1.14
	Dd	0.12	0.11	0.63	0.07	0.52	0.29
	Dd+Dp	*0.09*	*0.09*	0.13	0.06	0.19	0.11
	Dd+Du	**0.08**	*0.09*	0.55	0.08	0.47	0.25
	D3VO	**0.08**	*0.09*	*0.11*	*0.05*	0.19	**0.10**
S+I	VINS [175]	0.23	0.19	*0.11*	*0.10*	-	*0.17*
	OKVIS [40]	0.23	0.36	0.13	0.17	-	0.22
	Basalt [34]	**0.06**	*0.12*	**0.10**	**0.05**	-	**0.08**
	D3VO	*0.08*	**0.09**	*0.11*	**0.05**	-	**0.08**

8.5 Conclusion

We presented D3VO as a monocular VO method that enhances the performance of geometric VO methods by exploiting the predictive power of deep networks on three levels integrating predictions of monocular depth, photometric uncertainty and relative camera pose. To this end, we first introduced a novel self-supervised monocular depth estimation network which explicitly addresses the illumination change in the training set with predictive brightness transformation parameters. The network achieves state-of-the-art results on KITTI and EuRoC MAV. The predicted depth, uncertainty and pose are then incorporated into both the front-end tracking and back-end non-linear optimization of a direct VO pipeline. We

systematically evaluated the VO performance of D3VO on the two datasets. D3VO sets a new state-of-the-art on KITTI Odometry and also achieves state-of-the-art performance on the challenging EuRoC MAV, rivaling with leading mono-inertial and stereo-inertial methods while using only a single camera.

Acknowledgements We would like to thank Niclas Zeller, Lukas Köstler, Oleg Muratov and other colleagues from Artisense for their continuous feedbacks. Besides, we would like to thank Jakob Engel and Tao Wu for the fruitful discussions during the early stages of the project. Last but not least, we also would like to thank the reviewers and Klaus H. Strobl for their constructive comments.

Part IV

Conclusion and Outlook

Chapter 9

Summary

This thesis devised various techniques for visual-inertial odometry, SLAM, and relocalization. First, we focused on visual-inertial odometry, improving the marginalization procedure and IMU initialization. Afterwards we incorporated deep learning at various stages of optimization-based visual odometry. Lastly, we enhanced direct image alignment with deep features and a pose prediction network and applied them to relocalization across weathers and seasons.

Chapter 7: Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization[1]. We presented a visual-inertial odometry method which combines photometric optimization with IMU measurement errors. In order to enable immediate initialization of the visual-inertial system, we included scale and gravity direction as explicit optimization variables in the main system. To overcome the limitations of marginalization associated with evolving variable estimates, we proposed dynamic marginalization. Evaluations on the EuRoC dataset showed the importance of dynamic marginalization, and that our method outperformed the state of the art.

Chapter 4: DM-VIO: Delayed Marginalization Visual-Inertial Odometry. Following up on the previous chapter, we came up with delayed marginalization. Similar to dynamic marginalization, it improves the consistency of the marginalization prior, but it has a number of advantages over our previous technique: Delayed marginalization is not limited to a single one-dimensional variable, and retains more information. Additionally, it enables the proposed pose graph bundle adjustment, which captures the full visual uncertainty in an efficient optimization procedure. These contributions are combined into a multi-stage IMU initializer which is robust against long stretches of constant motion. Based on these techniques, a new visual-inertial odometry method called DM-VIO was developed. We provided extensive evaluations on three datasets covering flying drones, handheld, and automotive scenarios. They show that DM-VIO outperforms the state of the art in visual-inertial odometry, including popular stereo-inertial methods, while

[1] not examination-relevant

using only a monocular camera and IMU. For the benefit of the community, the full source code for DM-VIO, including a version for the Robotic Operation System (ROS), is available at `https://github.com/lukasvst/dm-vio`.

Chapter 8: D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry[2]. We demonstrated how to augment traditional visual-only odometry with learned knowledge. To this end, we proposed self-supervised networks which estimate monocular depth, photometric uncertainty, and the pose and affine brightness changes between two images. Depths, poses, and uncertainty were integrated into an optimization-based visual odometry system. The resulting method exceeds the state of the art in visual odometry on Kitti and EuRoC. On EuRoC it performs similar to the state-of-the-art visual-inertial odometry methods, while using only a single camera.

Chapter 5: GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. We developed GN-Net, a neural network which yields features tailored to direct image alignment. Compared to normal images, our deep features significantly improve the robustness against illumination and weather changes, enabling us to apply direct methods to relocalization. GN-Net is trained with the novel Gauss-Newton loss which we derived based on the probabilistic interpretation of the Gauss-Newton algorithm. To evaluate our method, we established a relocalization tracking benchmark based on CARLA and Oxford RobotCar. The experiments show that our method outperforms popular direct and indirect methods. Lastly, we demonstrated qualitatively, that our approach can be used to perform multi-weather relocalization.

Chapter 6: LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization. Based on observations of the typical behaviour of the Levenberg-Marquardt algorithm (applied to direct image alignment), we devised a complete loss formulation and point sampling strategy. We consider 4 cases which can happen during optimization and include a loss function for each. Our network LM-Net is trained with this loss formulation and significantly outperforms the previous GN-Net. Additionally, we proposed CorrPoseNet, a pose prediction network, which provides an initialization to the nonlinear optimizer. We carefully evaluated our approach on the CARLA and Oxford RobotCar relocalization tracking benchmark. Lastly, we quantitatively validated the effect of each contribution, and provided a qualitative relocalization demo.

[2] not examination-relevant

Chapter 10

Future Research

Extensions to DM-VIO using factor graphs. In order to ensure real-time performance of DM-VIO, the photometric bundle adjustment has to rely on highly-optimized code, similar to DSO. But unlike DSO, we have integrated the accelerated photometric residuals into a GTSAM factor graph, greatly improving flexibility. This allows to extend the system e.g. with more sensors like wheel odometry or GPS, by simply adding corresponding factors to the different factor graphs.

Another interesting direction is online calibration. Optimization of IMU extrinsics is already implemented and only needs to be activated. For temporal online calibration one can replace the IMU factors with the ones proposed in [177].

Improving SLAM / loop closure. DM-VIO performs pure odometry, but it can be extended to a full SLAM system. For this, some of the new techniques proposed in DM-VIO can be repurposed to improve the way loop closure and keyframe reactivation are performed in current systems.

For instance, pose graph bundle adjustment (PGBA) can replace pose graph optimization (PGO) during loop closure. Wheres PGO only inserts binary constraints between keyframes, PGBA models the full visual uncertainty, which might lead to improved accuracy of the resulting trajectory.

A drawback of marginalization is that by default it is incompatible with reactivation of observations. Systems like ORB-SLAM, which are able to reactivate keyframes and points after they have left the optimization window, do not use marginalization but instead have to fix variables. Delayed marginalization can be utilized to remove observations from the marginalization factor, and help to overcome this for small loop closures. Finding the optimal way to combine marginalization with keyframe / point reactivation is an interesting research topic.

Full learning-assisted SLAM. While the deep-learned features proposed in chapters 5 and 6 have been thoroughly evaluated, they have yet to be fully integrated into a live SLAM system. Such system can rely on images for frame-to-frame tracking while using LM-Net features for relocalization and loop-closure, and finding nearby images with techniques like NetVLAD [80].

Until now, our features have only been applied to direct image alignment (where the relative pose between two images is optimized with fixed geometry). However, they can also be utilized in the photometric bundle adjustment performed in the back-end of the SLAM system. By tightly integrating LM-Net, a SLAM system can perform bundle adjustment with a combination of live and map images, which might further improve accuracy.

Life-long learning. Learning-based methods are trained on particular datasets and assume that they will be applied in a reasonable vicinity of their training data. To address this, one can train on increasingly large datasets, or continue training after deployment. The techniques proposed in chapters 8, 5, and 6 can mostly be trained in a self-supervised fashion, e.g. using the output of the SLAM system. Hence, one could integrate them into a system which automatically performs retraining with successfully tracked data.

Learned scene representations. Despite utilizing learned knowledge in various parts of the pipeline, the proposed methods represent scenes as pointclouds during optimization. More sophisticated, potentially dense, representations have the potential to improve accuracy by allowing to model more aspects of the world, like patch geometry, normals and maybe even specularity and reflections. Deep learning can be applied to represent the scene in alternative ways, for instance with autoencoders or NeRFs [178]. This is an ongoing research topic [33], [179], [180] with impressive results for dense reconstruction. However, in terms of pose estimation accuracy, these new approaches have not yet reached traditional methods. Bridging this gap and fully leveraging the advantages of a dense representation remains an exciting research topic.

Part V

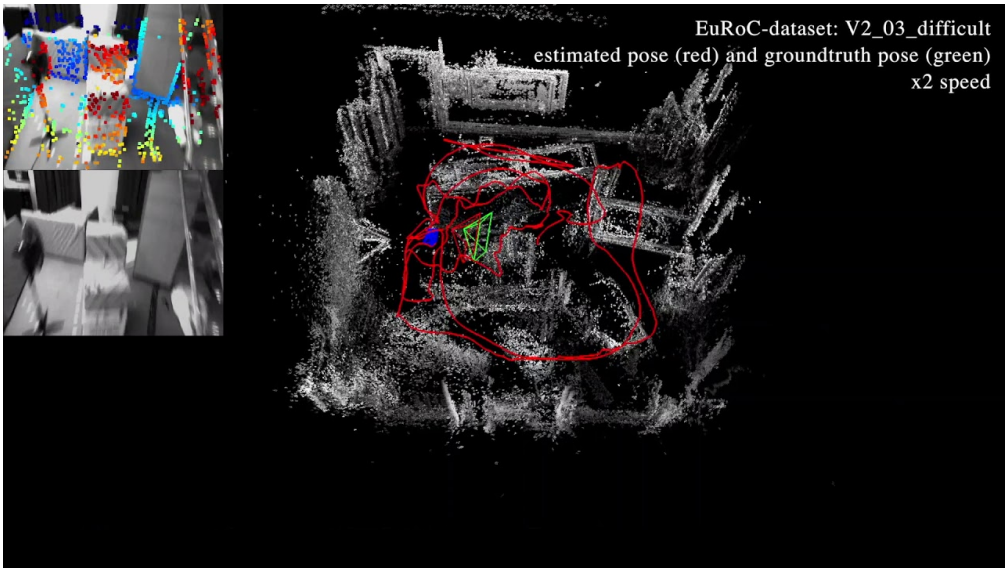
Appendix

Appendix A

Multimedia Material

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization

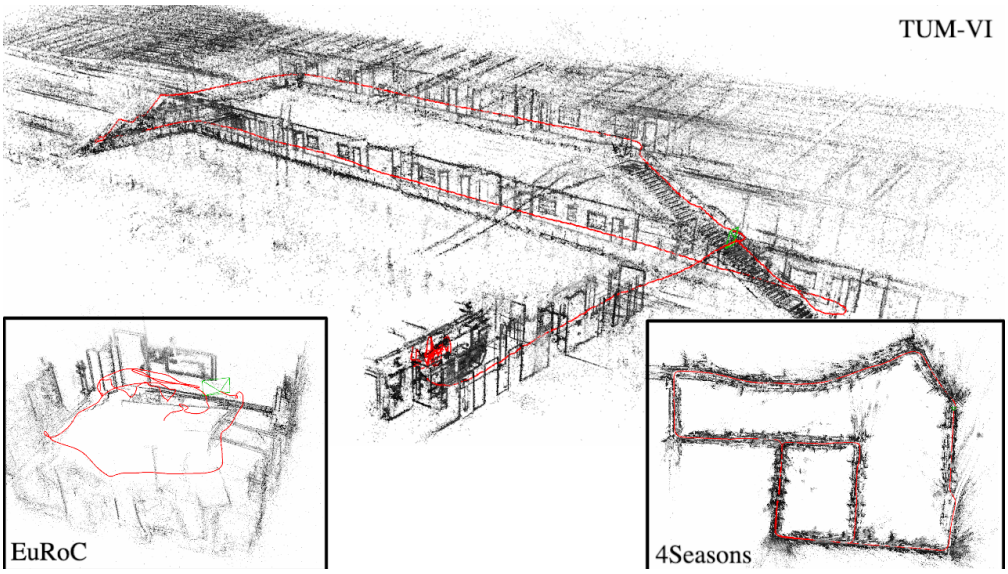
Results and animation explaining dynamic marginalization



https://youtu.be/GoqnXDS7jbA

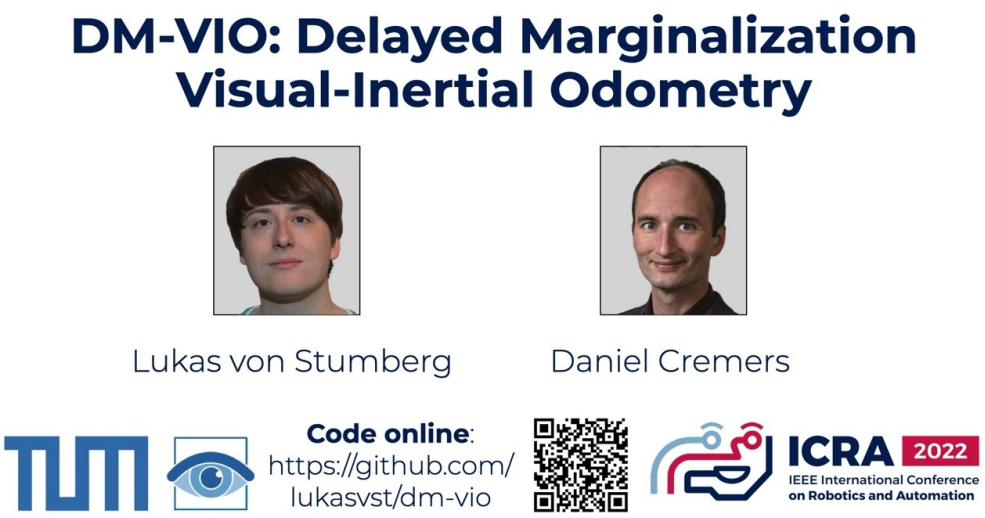
DM-VIO: Delayed Marginalization Visual-Inertial Odometry

Results video



https://youtu.be/7iep3BvcJPU

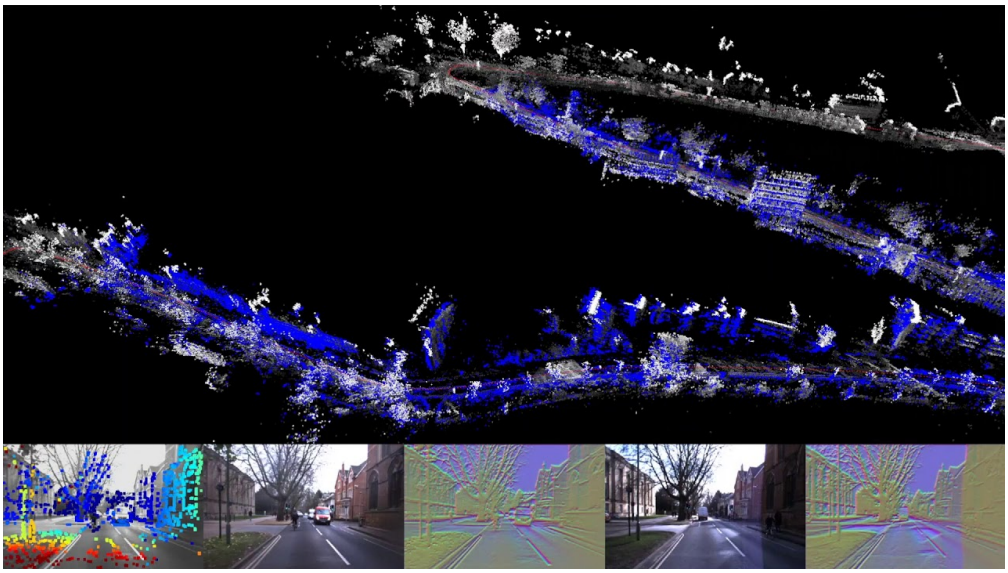
ICRA presentation video



https://youtu.be/pWKN7Afoirs

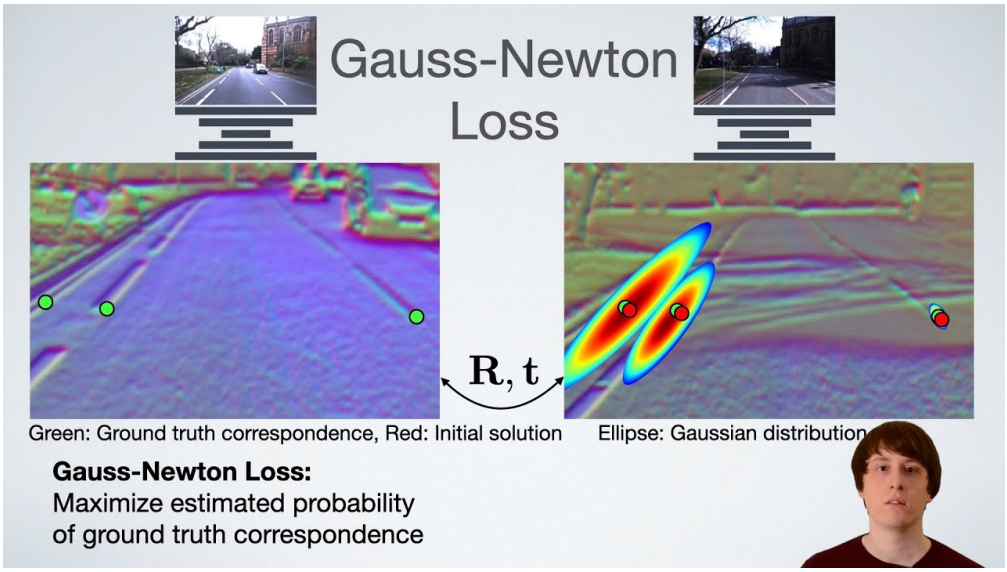
GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

Results and explanation of the Gauss-Newton loss



https://youtu.be/gcbKeKX2eiE

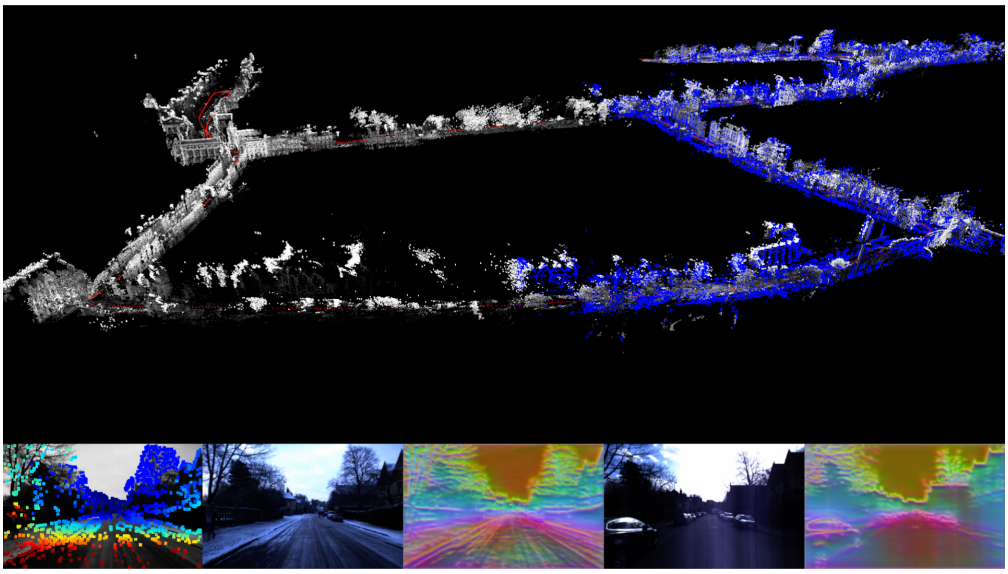
ICRA presentation video



https://youtu.be/q_uVb_o255o

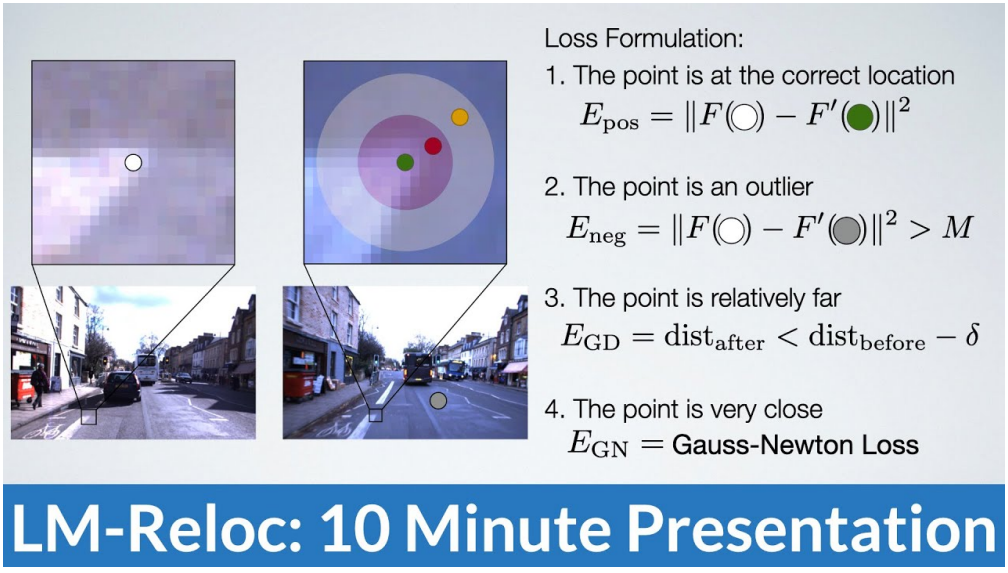
LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization

Relocalization demo



https://youtu.be/UHnbDEq7XQE

3DV presentation video



https://youtu.be/i7TyTwKD734

Appendix B

Open-Source Code and Datasets

DM-VIO: Delayed Marginalization Visual-Inertial Odometry

- https://github.com/lukasvst/dm-vio: Full source code of DM-VIO, including a live-version for the Realsense T265 camera.
- https://github.com/lukasvst/dm-vio-python-tools: Python tools for DM-VIO which can automatically download and prepare the datasets, run DM-VIO on all sequences to reproduce the results, and generate plots similar to the paper.
- https://github.com/lukasvst/dm-vio-ros: Robot Operating System (ROS) wrapper for DM-VIO, which allows it to run on live data.

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

- https://vision.in.tum.de/gn-net: Relocalization tracking benchmark with CARLA and Oxford RobotCar sequences.
- https://github.com/Artisense-ai/GN-Net-Benchmark: Tools for the benchmark.

Appendix C

Supplementary Materials

In the following, we include the supplementary materials for the three examination-relevant Chapters 4, 5, and 6.

Additional evaluations for DM-VIO: Delayed Marginalization Visual-Inertial Odometry

Lukas von Stumberg and Daniel Cremers

We provide an ablation study on different parts of the IMU initializer in section I, an ablation study on the impact of the dynamic photometric weight in section II, and runtime results in section III.

I. Ablation IMU Initializer

We perform an ablation study on various parts of the IMU initializer. In particular, we compare to the following three baselines, which build on top of each other, meaning that everything removed in baseline n is also removed in baseline $n+1$.

1) We remove the Reinitialization and the Marginalization replacement, corresponding to the orange and the yellow boxes in Fig. 3 of the main paper. Note that the scale is still optimized in the main system after initialization.
2) No readvancing: We do not replace the marginalization prior of the main graph after the first initialization. This means that in the purple box "Initialize" in Fig. 3 of the main paper we only replace the values and not the marginalization prior of the main graph. Instead we add a constant prior on the initial scale, which is necessary for the scale to not immediately diverge afterwards.
3) No PGBA: We remove the "PGBA IMU Init" and directly initialize with the result of the Coarse IMU Init.

The experiments are performed in non-realtime mode in order to not be dependent on the particular machine.

The results on the 4Seasons dataset are shown in Fig. S1. With each ablation the result becomes significantly worse, showing that all parts of the method contribute to the results. In particular we observe that the proposed delayed marginalization is very important, as it is the foundation of both the pose graph bundle adjustment (PGBA) and the readvancing (and subsequently also the marginalization replacement).

For completeness we should mention that on TUM-VI and EuRoC the contributions do not bring a significant performance improvement, as both datasets are much less challenging for the IMU initialization.

II. Ablation Dynamic Photometric Weight

We provide ablation studies showing the effect of the dynamic photometric weight proposed in the main paper. For this we disable the dynamic part and always use the constant weight $W = \lambda$ (of course λ is set to the same value as for the other results). In Fig. S2a we compare to the results shown in the main paper on TUM-VI, both runs are in realtime mode.

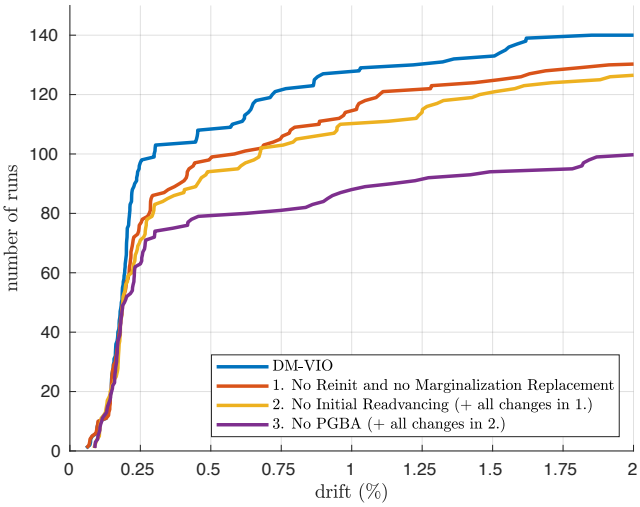
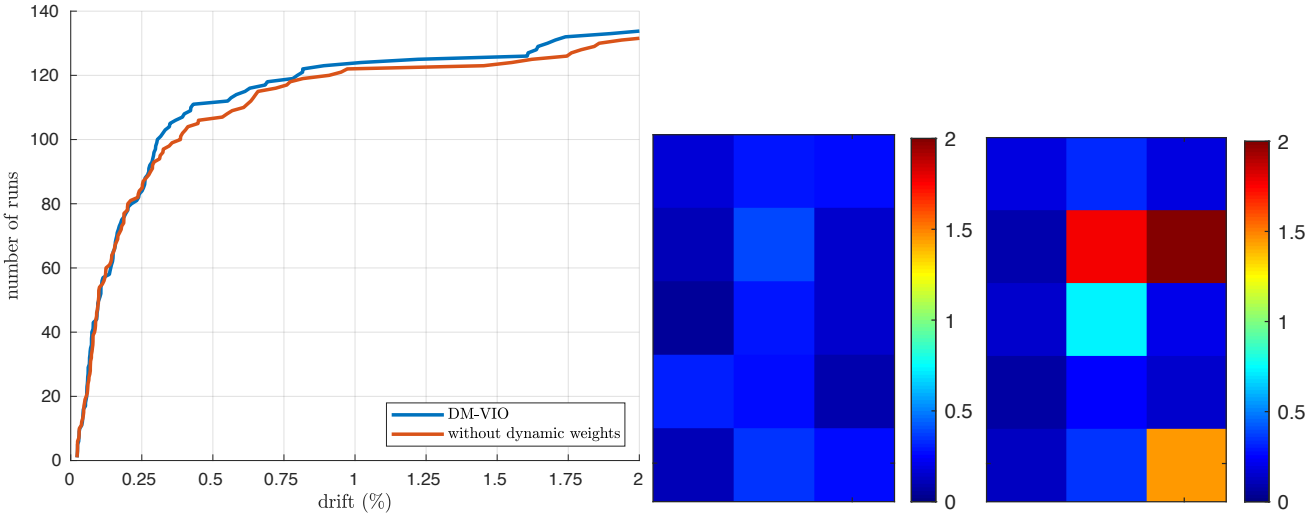


Fig. S1: Ablation study on various parts of the IMU initializer on the 4Seasons dataset (non-realtime). Removing the reinitialization and the marginalization replacement (1.) makes a significant difference, as the marginalization prior can become inconsistent if the initial scale estimate is off. Removing also the initial marginalization replacement powered by readvancing (2.) further deteriorates the performance. Removing the pose graph bundle adjustment and only using the Coarse IMU init (3.) makes the biggest difference. This ablation shows the significance of delayed marginalization which is the foundation of PGBA, readvancing and marginalization replacement.

It can be seen that the dynamic weights improve the overall robustness of the method. The effect is most visible on the TUM-VI slides (Fig. S2b, S2c), where the version without dynamic weights does not work well in 4 of the 15 runs whereas our method works well in all runs. The dynamic weighting is designed for situations where the image quality gets really bad. On the other datasets, significant degradation of image quality is rare, hence there is only a marginal difference for them.

III. Runtime

We perform extensive runtime analysis on the same MacBook Pro 2013 (i7 at 2.3GHz) that was used for generating the results in the main paper. For each part of the method we save the mean time and standard deviation it takes while processing a sequence. We run our method in real-time mode 10 times on each sequence of EuRoC dataset, resulting in 110 recorded means and standard deviations for each part.



(a) Cumulative error plot.

(b) DM-VIO on the slides.

(c) Without dynamic weights on the slides

Fig. S2: Realtime results on the TUM-VI dataset with and without the dynamic photometric weights. a) Cumulative error plot. The dynamic weight provides a noticeable improvement in robustness. Only few sequences contain hard enough images to trigger the dynamic weight, hence the overall improvement can appear small. However for these hard sequences, success rate is increased. b) and c): Each colored square represents the drift (%) for one of the 5 runs (rows) on the three slides sequences (columns). With dynamic weights our method works well for all 30 runs whereas the version without them fails once and accumulates a large drift for three executions.

TABLE S1: We save runtime statistics for different parts of the method and show the mean over all 110 runs on the EuRoC dataset (10 times on each sequence). Top 2 sections: Runtime for the two main threads. Bottom 3 sections: Runtime for the parts of the IMU initializer, which are performed occasionally. Note that the overhead introduced by our initializer is small: The only regular overhead is the delayed marginalization, taking 0.44ms which amounts to 0.8% of the keyframe operations. Both, the Coarse IMU initializer and the PGBA initializer perform most operations in a separate thread when running. The only relevant overhead in the keyframe thread is after a successfull PGBA and during a marginalization replacement.

Component	Part	Mean time (ms) Mean of 110 runs	Standard deviation (ms) Mean of 110 runs
Coarse Tracking Performed for every frame.	Total	10.34	10.16
	Direct Image Alignment	4.70	1.31
	Build Image Pyramid	4.78	0.52
Keyframe Operations Performed for every keyframe.	Total	53.67	6.89
	Bundle Adjustment	26.49	5.57
	Create Candidate Points	5.83	0.38
	Trace candidate points	4.37	1.83
	Activate new points	3.56	0.77
	Keyframe Marginalization (full)	3.24	0.5
	— Drop residuals, recompute adjoints, etc.	— 1.71	— 0.39
	— Marginalize main graph	— 0.61	— 0.10
	— **Delayed marginalization**	— 0.44	— 0.20
Coarse IMU Init	Runtime in separate thread	9.08	5.63
	Overhead Coarse Tracking thread	0.05	0.04
PGBA (Re-)init	Runtime in separate thread	74.11	24.77
	— Optimization	— 54.72	— 23.40
	— Readvancing	— 7.61	— 1.37
	Overhead Keyframe thread when active	0.17	0.04
	Overhead Keyframe thread after success	14.50	8.52
	— Include KFs added while PGBA was running	— 13.15	— 8.52
	— Readvancing of the newly added KFs	— 1.35	— 0.19
Marginalization Replacement	Total (Keyframe Thread)	21.02	8.98
	Build Graph	1.15	0.45
	Readvance	19.87	8.97

We present the mean mean runtime and the mean standard deviation over the 110 runs in Table S1.

Even on the relatively old machine the overall runtimes are quite fast. The tracking could run at 97 FPS on average and the keyframe processing could run at 19 FPS. Note that the dataset is recorded at only 20 FPS, and to work well our method needs to process 5-6 keyframes per second.

The only regular overhead introduced by our initializer is the delayed marginalization, which amounts to 0.8% of the total processing time of each keyframe. The Coarse IMU initializer and the PGBA both run in separate threads when active. We do note that after a successful PGBA, some processing has to be done in the keyframe thread in order to include the keyframes, which have been bundle adjusted while the PGBA was running. Similarly the marginalization replacement currently runs in the keyframe thread. However both of these parts are comparably fast (14.50ms and 21.02ms respectively), and are performed only a few times during every run.

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization – Supplementary Material –

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Qadeer Khan[1,2] Daniel Cremers[1,2]
[1]Technical University of Munich [2]Artisense

Abstract

In this document, additional information complementing the original paper is given. Details of our relocalization tracking *benchmark are also described. Furthermore, we provide results of some additional experiments evaluating the performance of our method to indoor and homogeneous scenes. Lastly, additional details on the network architecture used for this work is also provided. The video and the benchmark dataset can be found at* `https://vision.in.tum.de/gn-net`.

A. Details on the Evaluation Benchmark

This section describes the data included in our benchmark which is used for *relocalization tracking*. It contains simulated data created with the CARLA [2] simulator. It also provides Lidar aligned real-world sequences from the Oxford RobotCar dataset [4]. The CARLA benchmark is constructed with the stable version 0.8.2 of the simulator. We add ground-truth poses and camera intrinsics for all images collected under different lighting and weather conditions. The dataset presents the challenge of relocalization against different weather conditions and poses. We have collected data from 6 different cameras. The positions and orientations of the cameras are given in Table 1. A subset of the different positions and orientations of 3 of the 6 cameras are shown in Figure 1 for a certain time step. The cameras are mounted relative to the ego-vehicle. For each camera, 500 images are collected. For training, validation, and testing sets, 9 different sequences for each are recorded under three different weather conditions. The conditions and the overall statistics of the training, validation, and testing datasets are given in Table 2. Along with the camera images, we also provide access to their corresponding dense depth maps and semantic labels. The data is generated by driving around *Town1* of the CARLA simulator.

Relocalization tracking: For each one weather sequence, we have created a *relocalization.txt* and for each all weathers sequence a *relocalization_other_weathers.txt* file. The difference between the two relocalization files is that the latter one contains cameras to be relocalized against different sequences and hence different weather conditions. Each row of the relocalization file is arranged as follows: Current camera index, camera to be localized against, and relative pose between these cameras. The relative pose between the two cameras is in the coordinate system of the left camera. The *stereocalibration.txt* provides the relative pose information between Cam0 and Cam1. Furthermore, *transforms.json* contains the extrinsic parameters of all cameras along with *camera_intrinsics.json*. We have withheld the ground truth data for the testing sets which can be evaluated by submitting the relocalized camera poses to our servers which shall be established upon acceptance.

Camera Id	X	Y	Z	Roll	Pitch	Yaw
Cam0	2.2	0.0	1.3	8	0	0
Cam1	2.2	0.5	1.3	8	0	0
Cam2	2.2	-0.5	1.3	8	0	0
Cam3	2.5	0.5	2.3	4	-20	0
Cam4	2.6	-0.7	2.3	-10	27	0
Cam5	3.0	-1.2	2.1	20	14	0

Table 1. This table describes the camera positions and orientations mounted relative to the ego-vehicle.

condition	#cameras	#images individual	sequences	total
WetNoon	6	500	3	9,000
SoftRainNoon	6	500	3	9,000
WetCloudySunset	6	500	3	9,000
total	-	1,500	9	27,000

Table 2. This table provides a summarized information about the proposed CARLA benchmark. It describes the weather scenarios under which data was collected, the number of cameras, sequences and the images for each.

Note that only for the training and validation sets, we additionally provide dense depth maps and the semantic segmentations for all the images. In order to understand the general structure of the benchmark, the folder tree is given

*These authors contributed equally.

below.

```
benchmark_sample
├── episode_000
│   ├── relocalization.txt
│   ├── relocalization_other_weathers.txt
│   ├── transforms.json
│   ├── calibs
│   │   ├── camera.txt
│   │   └── stereocalibration.txt
│   ├── CameraDepth0
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── ...
│   ├── .....
│   │   └── ....
│   ├── CameraDepth5
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── .....
│   ├── CameraRGB0
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── ...
│   ├── .....
│   │   └── ....
│   ├── CameraRGB5
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── .....
│   ├── CameraSemSeg0
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── ...
│   ├── .....
│   │   └── ....
│   ├── CameraSemSeg5
│   │   ├── image_00000.png
│   │   ├── image_00001.png
│   │   └── .....
└── episode_001
    ├── relocalization.txt
    ├── ......
    │   └── .......
    └── .......
```

Our benchmark for the Oxford RobotCar [4] sequences follows the same structure as that for CARLA described above, but only utilizes images recorded from a stereo camera setup rather than having 6 different cameras. The resulting point clouds from oxford sequences are aligned with the global registration followed by ICP alignment using the implementation of Open3D [7]. This alignment was performed for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*.

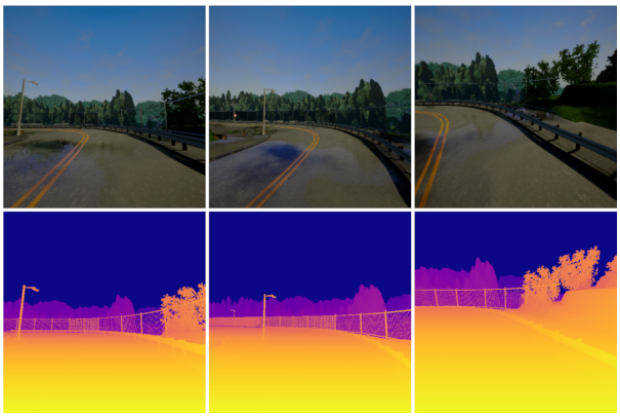


Figure 1. This figure shows a subset of 3 of the 6 camera images (top row) from the same time step along with the corresponding dense depth map (bottom) rendered from the simulation engine. The cameras are oriented at different positions and orientations with respect to each other. This provides 6 DOF. Multiple cameras are used to enhance the variety in the dataset for training a robust deep feature prediction network. However, it is important to mention here that the correspondences used to train our models were determined from the point clouds using DSO [6].

B. Additional Experiments

In these supplementary experiments, we show that our method significantly improves the robustness for large-baseline tracking even when there are no weather/lighting changes involved. This is done by evaluating on our CARLA benchmark with only one weather, as well as on the indoor EuRoC dataset [1].

B.1. CARLA results for different weathers

As mentioned in the main paper we show results on our CARLA benchmark for relocalization between different weather conditions in Figure 2. Figure 3 shows feature matrices produced by our model, showing similar feature maps even for images with differing lighting/weather conditions.

B.2. Evaluation of robustness to large baselines/low frame rates

We demonstrated in the main paper that our method greatly improves robustness and accuracy for tracking across different weathers.

However, even when tracking images with similar lighting conditions our deep features greatly improve tracking

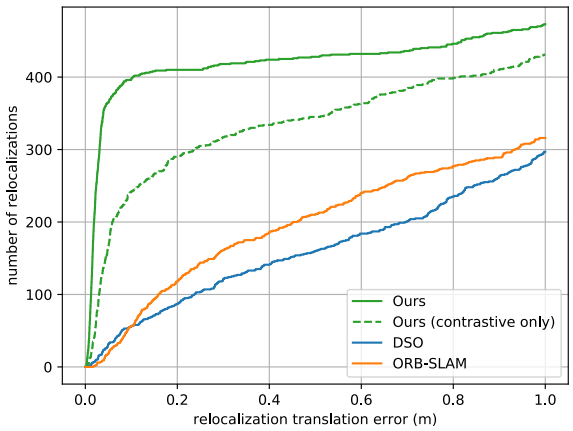


Figure 2. This figure shows the cumulative relocalization accuracy for tracking against different weathers on the CARLA benchmark. ORB-SLAM is more robust to changes in lighting, and weather, whereas DSO shows the worst performance. By utilizing our trained deep descriptors, we are able to outperform both methods by a large margin. Notice that our novel Gauss-Newton loss has a large impact as the model trained only with the contrastive loss performs significantly worse.

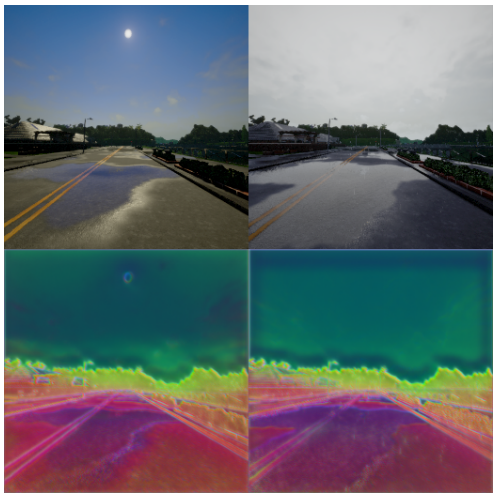


Figure 3. This figure shows images and their corresponding feature maps predicted by our GN-Net for two different weather conditions included in our CARLA benchmark. Despite shadows, raindrops, and water puddles the feature maps are very much similar. Note that the feature maps are displayed via a lower-dimensional representation using PCA.

performance for large baselines, which we will show in this section on the CARLA and the EuRoC datasets. For all experiments in this section we have changed the following two hyperparameters: the vicinity for the GN-Loss is changed from 1 to 3 pixels and the second term of the Gauss-Newton loss is weighted by $2/7$.

CARLA: We use the first three sequences of the training, validation, and test set provided by our benchmark, which all capture the same weather condition. As these sequences do not contain substantial illumination changes any differences in the performance will mainly show the general accuracy and robustness of the methods. Again, all hyperparameter tuning is only performed on the training and validation set. The cumulative relocalization accuracy for tracking against the same weather for the 3 evaluated methods is shown in Figure 4. While DSO is more accurate, the indirect ORB-SLAM system has higher robustness. In contrast, our GN-Net is not only as accurate as DSO, but also more robust than ORB-SLAM. This is because of the larger convergence basin of the features maps created by our network.

EuRoC dataset with low framerates: For this experiment we use a more traditional metric and evaluate on the challenging EuRoC MAV dataset [1]. We run each method on the 11 sequences of the dataset and evaluate the absolute trajectory error of the estimated poses against the ground-truth. Note that in this experiment no relocalization tracking is involved. For ORB-SLAM we have disabled loop-closures and relocalization to enable a fair comparison with the other pure odometry methods. Stereo DSO is used without modification.

For our method we have modified the normal frame-to-frame tracking performed by the *coarse tracker* of Stereo DSO to use deep features instead. As our features have a larger convergence basin than normal images, we expect this to improve the robustness of the tracking against large baselines.

In order to evaluate the performance of our method on all the 11 sequences, we split the sequence into 2 sets, while training 2 different models. Set A contains the first 6 of the 11 sequences, while set B comprises of the remaining 5. The first model is trained with set A and evaluated on set B. The second model is trained with set B and evaluated on set A. The final evaluation reported is the combination of the evaluation results from these 2 models. This way we are able to cover all the 11 sequences in our evaluation.

In order to evaluate the performance of the methods with low-framerate cameras (thus including larger-baselines) we subsample the frames included by skipping n frames for each frame that is used. Each method is run 3 times for all 11 sequences, for each $n \in [0, 7]$. For $n = 7$, e.g. this means only every 8th image is used, simulating a frame-rate of 2.5 Hz. The results are shown in Figure 5, again demonstrating that our features significantly improve the robustness of direct SLAM methods.

C. Network Architecture Details

We adopt a similar network architecture as the U-Net model [5] as seen in Figure 6. What is different is that we change the decoder part such that our feature maps can leverage a multiscale hierarchical feature pyramid which allows propagating information from coarser to finer levels of the pyramid. For the encoder part, we followed the conven-

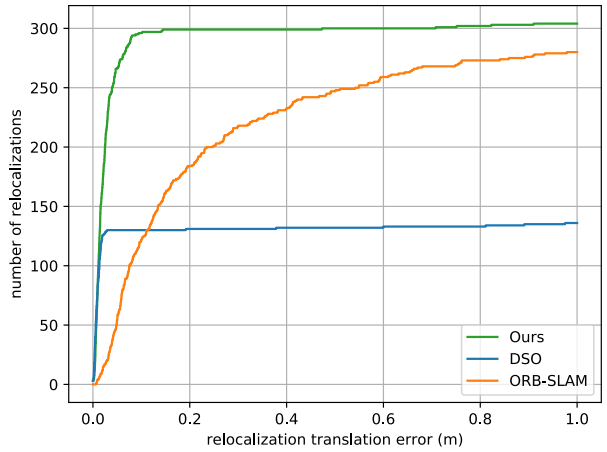


Figure 4. This figure shows the cumulative relocalization accuracy for tracking against the *same weather*. ORB-SLAM tracking is less accurate but more robust than normal direct image alignment. Our approach outperforms both of them. This shows that our approach improves not only robustness to challenging lighting situations, but also to large-baseline tracking.

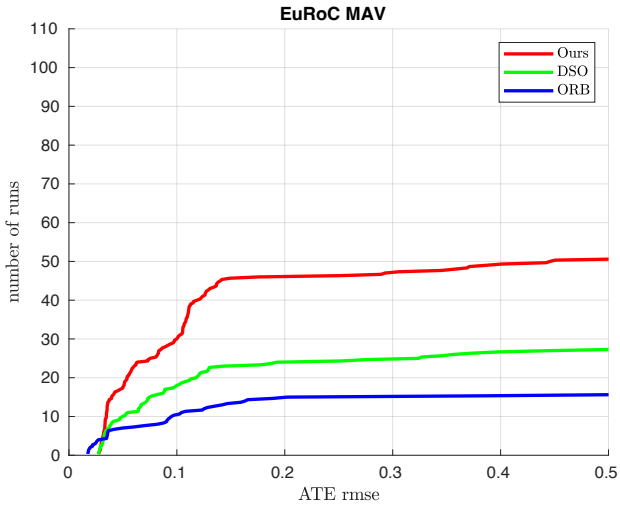


Figure 5. A cumulative plot of the absolute trajectory error (RMSE) on the EuCoC dataset for different numbers of skipped frames. In this experiment, we have replaced the frame-to-frame tracking used in Stereo DSO with our deep feature matrices. We then evaluate the accuracy of the estimated trajectory when running all sequences of the dataset with different frame-rates ranging between 20Hz and 2.5 Hz We compare to normal Stereo DSO and ORB-SLAM without loop-closures. Note that our models for this experiment are trained entirely self-supervised, yet improving DSO robustness almost by a factor of two.

tion of [5]. The encoder part consists of four downsampling blocks, for which each of them uses a 2×2 max pooling operation with stride 2, followed by a convolutional block which is executed two times. The convolutional block applies a 3×3 convolution with padding of 1, followed by batch normalization, and an exponential linear unit (ELU). At each downsampling step, the number of channels is doubled. The number of feature channels is set to 64.

Decoder modification: We change the decoder part of the architecture from the default U-Net architecture in the following way. Beginning from the coarsest level, we upsample (with bilinear interpolation) the feature maps by 2 and concatenate those feature maps with the feature map of the higher level. After this, we apply D number of 1×1 convolution kernels to make the filter maps of the same channel size. This is done in an iterative fashion until the finest level. This results in the final feature pyramid map representation which we use for deep direct SLAM. The feature map sizes are described in Table 3.

Level	S
3 (coarsest)	$D \times H/8 \times W/8$
2	$D \times H/4 \times W/4$
1	$D \times H/2 \times W/2$
0 (finest)	$D \times H \times W$

Table 3. The hyperparameter setting for our network architecture. Level: level of the network. S : size of the feature map. H, W : height and width of the image, respectively.

For all experiments, we use $D = 16$ channels as a feature output vector size. For all trainings, we use the Adam optimizer [3] for optimization with a learning rate of 10^{-6} and a weight decay of 10^{-3}. For the correspondence contrastive loss term, we set the margin $M = 1$. For the Gauss-Newton loss, we set the maximum distance of the start point to the correct point to 1 pixel for all experiments in the paper and to 3 pixels for the experiments in this supplementary material. All steps of the optimizer use a single image pair as input to the network. Each pair of images fed to the Siamese network architecture has a number of positive correspondences and for each of them, a negative correspondence is randomly sampled. For CARLA the input image size is $W = 512$, $H = 512$ and for Oxford RobotCar it is $W = 640$, $H = 480$.

D. Implementation Details

Coupling of GN-Net with DSO: DSO contains two components where images are used for pose estimation. In the Bundle Adjustment (BA) the pose of 8 keyframes together with the inverse depth of all active points is optimized. For this optimization, a very good initialization is assumed. The *coarse tracking* is performed for every image and optimizes the 6DOF pose between the latest keyframe and the current frame together with two affine brightness parameters a and b which represent a brightness transformation between the two images. This is done using normal direct image alignment in a pyramid scheme.

We have adopted the coarse tracker to be able to use our

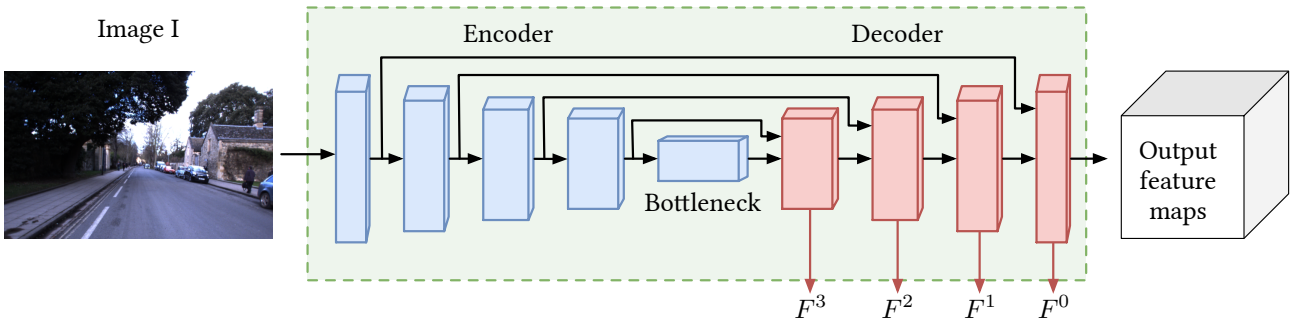


Figure 6. Overview of one branch of the Siamese network architecture. Each branch is a modified U-Net [5] architecture which receives as an input an image and predicts multi-scale feature maps $[F^0, F^1, F^2, F^3]$. The multi-scale feature maps from the decoder network of both branches are then passed and used by DSO. Note that the weights between the two branches are shared.

multi-channel feature maps as an input instead of images. Notice that we directly input *all* the pyramid levels created by our network instead of downscaling the image as it is done in DSO. This modified coarse tracker is then used for relocalization tracking.

Notably, the network only takes a single image as an input to create the feature maps. This means that the runtime of the inference scales linearly with the number of images involved. Therefore it would be possible to also use the features for the BA, although this has not been done in this specific work. Our network extracts features at over 8Hz. In principle this would be enough for real-time operation as we perform relocalization only on keyframes which arrive at less than 5Hz usually.

Details of the relocalization demo: Our method works by performing relocalization tracking as in the previous experiment, which yields a solution for the transformation between the current world and the map $T_{\text{map}}^{\text{world}}$. The results for this transform are optimized in a factor graph with a fixed random walk between successive solutions. As again we do not consider finding candidate images, we supply the first corresponding image in the map to the methods. From there on no supervision signal is given. After bootstrapped with the first image our method finds the next relocalization candidates by determining the keyframe in the map with the minimum distance to the current image, which can be computed using the current solution for the transform $T_{\text{map}}^{\text{world}}$.

E. Additional proofs

Here, we show why the linear system used in the Gauss-Newton algorithm (Equation (6) of the main paper) also defines a Gaussian probability distribution.

The Gauss-Newton algorithm assumes a Taylor expansion of the energy:

$$E(\mathbf{x}') \approx E(\mathbf{x}_0) + \mathbf{b}^T(\mathbf{x}' - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x}' - \mathbf{x}_0)^T\mathbf{H}(\mathbf{x}' - \mathbf{x}_0) \tag{1}$$

The optimal solution according to this approximation can be obtained by finding the point where the derivative of the energy is 0

$$0 \stackrel{!}{=} \frac{dE}{d\mathbf{x}} = \mathbf{b} + \mathbf{H}(\mathbf{x}' - \mathbf{x}_0) \tag{2}$$

$$\mathbf{x}' = \mathbf{x}_0 - \mathbf{H}^{-1} \cdot \mathbf{b} \tag{3}$$

As the taylor expansion is performed around $\mathbf{x_0}$ we from now on set $\mathbf{x} = \mathbf{x}' - \mathbf{x_0}$.

On the other hand, we can try to find the the maximum point of a Gaussian distribution, by minimizing the negative log-likelihood (Equation (7-8) in the main paper):

$$E(\mathbf{x}) = -\log f_X(\mathbf{x}) = \tag{4}$$

$$\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) = \tag{5}$$

$$\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}^T - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \tag{6}$$

$$\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right)\boldsymbol{\mu} \tag{7}$$

Now we set $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\mu} = -\mathbf{H}^{-1}\mathbf{b}$. Then

$$(6) = \frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}^T - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \text{const} = \tag{8}$$

$$\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{b}^T\mathbf{x} = (1) \tag{9}$$

This equality shows that our the linear system in the Gauss-Newton step also represents a Gaussian probability distribution with covariance $\mathbf{H}^{-1}$ and a mean at the solution of the Gauss-Newton step.

References

[1] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC Micro Aerial Vehicle Datasets. *IJRR*, 35(10), 2016.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *CoRL*, 2017.

[3] D. P. Kingma and J. L. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[4] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1), 2017.

[5] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[6] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *ICCV*, 2017.

[7] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv 2018*, 2018.

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization: Supplementary Material

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Nan Yang[1,2] Daniel Cremers[1,2]
[1] Technical University of Munich [2] Artisense

A. Video

As mentioned in the paper, we provide a video of the qualitative relocalization demo, which is available at https://vision.in.tum.de/lm-reloc.

B. Network Architecture

CorrPoseNet. The CorrPoseNet takes 2 images (I and I') as the input and outputs the relative pose R, t between those images. The overall network architecture of the CorrPoseNet is depicted in Figure 1. The convolutional blocks consist of in total 9 convolutional layers followed by ReLU activations. The architectural details of the convolutional blocks are listed in Table 1. The correlation layer which takes the output of the convolutional blocks as input is described in the main paper. The correlation layer is followed by the regression block which regresses the relative pose. The layers of the regression block are listed in Table 2. The output of the network is the rotation R as Euler angles and translation t.

Table 1: Network architecture and parameters of the convolutional blocks. **k** denotes kernel size, **s** stride, and **p** padding.

Convolutional blocks						
layer	**in-chns**	**out-chns**	**k**	**s**	**p**	**activation**
conv0	3	16	16	2	3	ReLU
conv1	16	32	5	2	2	ReLU
conv2	32	64	3	2	1	ReLU
conv3	64	64	3	1	0	ReLU
conv4	64	128	3	2	2	ReLU
conv5	128	128	3	1	1	ReLU
conv6	128	256	3	2	1	ReLU
conv7	256	256	3	1	1	ReLU

LM-Net. We adopt U-Net [1] as the encoder of LM-Net. However, we change the decoder part of the architecture in the following way. Starting from the coarsest level, we upsample (with bilinear interpolation) the feature maps by 2 and concatenate those feature maps with the feature map of the higher level. This is followed by 1×1 convolutional filters. This procedure is repeated 4 times. This results in the feature pyramid maps as described in Table 3.

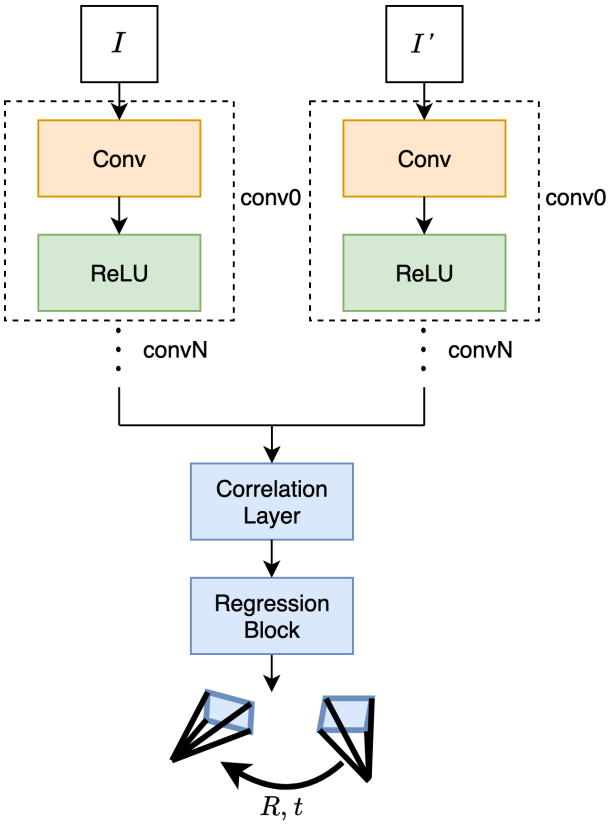
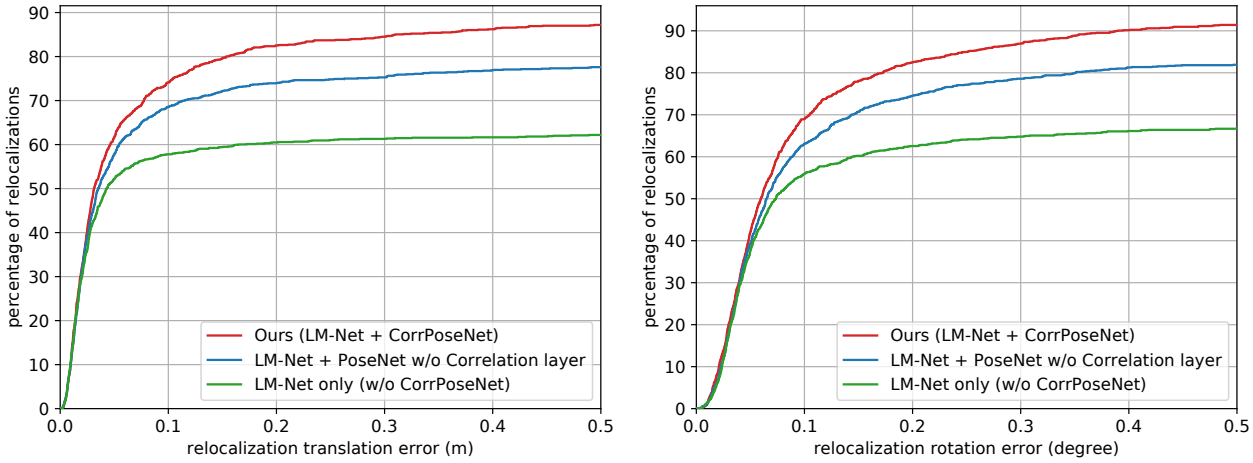


Figure 1: Network architecture of CorrPoseNet.

C. Ablation Study Correlation Layer

We demonstrate the impact of the Correlation layer in the proposed CorrPoseNet. We compare it to a simpler pose estimation network where the correlation and regression layers are replaced with two 1×1 convolutions with 3 output-channels each, which directly regress rotation and Euler angles. This simpler PoseNet has one more convo-

*Equal contribution.



(a) Translation error.

(b) Rotation error.

Figure 2: Cumulative error plot for relocalization on the CARLA relocalization benchmark validation data [2]. It can be seen that the correlation layer in CorrPoseNet has a large impact on the performance.

Table 2: Network architecture and parameters of the regression block. **k** denotes kernel size, **s** stride, and **p** padding. $N_c = 256$ denotes the input channels for the CARLA model, and $N_o = 260$ denotes the input channels for the Oxford model, respectively.

Regression block						
layer	**in-chns**	**out-chns**	**k**	**s**	**p**	**activation**
conv0	N_c / N_o	128	7	1	0	ReLU
BN	128	128	-	-	-	
conv1	128	64	5	1	0	ReLU
BN	64	64	-	-	-	
FC	2304	6	-	-	-	

Table 3: Output of the decoder of LM-Net. H, and W denote height and width of the feature maps.

Decoder layer	**Output size**
F_1	$16 \times H/8 \times W/8$
F_2	$16 \times H/4 \times W/4$
F_3	$16 \times H/2 \times W/2$
F_4	$16 \times H \times W$

lutional block conv8 with 512 output channels, kernel size 3, stride 2, and padding 1. Otherwise the network architecture and parameters are the same as for CorrPoseNet. The results on the CARLA validation data are shown in Figure 2. Even the simpler pose estimation network (PoseNet w/o Correlation layer) improves the result over using identity as an initialization for the direct image alignment (LM-Net only). However, utilizing the correlation layer significantly boosts the performance.

Table 4: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the Oxford validation sequences. Our data augmentation (which warps the images using random poses) improves both rotation and translation error.

Method	t_{AUC}	R_{AUC}
Ours	**80.45**	**65.11**
Ours w/o data augmentation	80.15	64.58

D. Ablation Study Oxford Data Augmentation

We show the impact of the data augmentation for the Oxford RobotCar Relocalization benchmark, where we warp the images to different poses using dense depths in Table 4. It can be seen that the proposed augmentation improves translation and rotation error on the validation data.

References

[1] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. 1

[2] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. *RA-L*, 5(2):890–897, 2020. 2

Appendix D

Original Publications

The publications included in the main part (Chapters 7 - 6) have an adapted layout. In this appendix, we include the original publications. Following the reprint conditions of IEEE, these are the accepted versions. The disclaimers for the papers can be found on the first page of the respective Chapter in the main section.

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization

Lukas von Stumberg[1], Vladyslav Usenko[1], Daniel Cremers[1]

***Abstract*— We present VI-DSO, a novel approach for visual-inertial odometry, which jointly estimates camera poses and sparse scene geometry by minimizing photometric and IMU measurement errors in a combined energy functional. The visual part of the system performs a bundle-adjustment like optimization on a sparse set of points, but unlike key-point based systems it directly minimizes a photometric error. This makes it possible for the system to track not only corners, but any pixels with large enough intensity gradients. IMU information is accumulated between several frames using measurement preintegration, and is inserted into the optimization as an additional constraint between keyframes. We explicitly include scale and gravity direction into our model and jointly optimize them together with other variables such as poses. As the scale is often not immediately observable using IMU data this allows us to initialize our visual-inertial system with an arbitrary scale instead of having to delay the initialization until everything is observable. We perform partial marginalization of old variables so that updates can be computed in a reasonable time. In order to keep the system consistent we propose a novel strategy which we call "dynamic marginalization". This technique allows us to use partial marginalization even in cases where the initial scale estimate is far from the optimum. We evaluate our method on the challenging EuRoC dataset, showing that VI-DSO outperforms the state of the art.**

I. Introduction

Motion estimation and 3D reconstruction are crucial tasks for robots. In general, many different sensors can be used for these tasks: laser rangefinders, RGB-D cameras [14], GPS and others. Since cameras are cheap, lightweight and small passive sensors they have drawn a large attention of the community. Some examples of practical applications include robot navigation [25] and (semi)-autonomous driving [11]. However, current visual odometry methods suffer from a lack of robustness when confronted with low textured areas or fast maneuvers. To eliminate these effects a combination with another passive sensor - an inertial measurement unit (IMU) can be used. It provides accurate short-term motion constraints and, unlike vision, is not prone to outliers.

In this paper we propose a tightly coupled direct approach to visual-inertial odometry. It is based on Direct Sparse Odometry (DSO) [6] and uses a bundle-adjustment like photometric error function that simultaneously optimizes 3D geometry and camera poses in a combined energy functional. We complement the error function with IMU measurements. This is particularly beneficial for direct methods, since the error function is highly non-convex and a good initialization is important. A key drawback of monocular visual odometry is that it is not possible to obtain the metric scale of the environment. Adding an IMU enables us to observe the scale. Yet, depending on the performed motions this can take infinitely long, making the initialization a challenging task. Rather than relying on a separate IMU initialization we include the scale as a variable into the model of our system and jointly optimize it together with the other parameters.

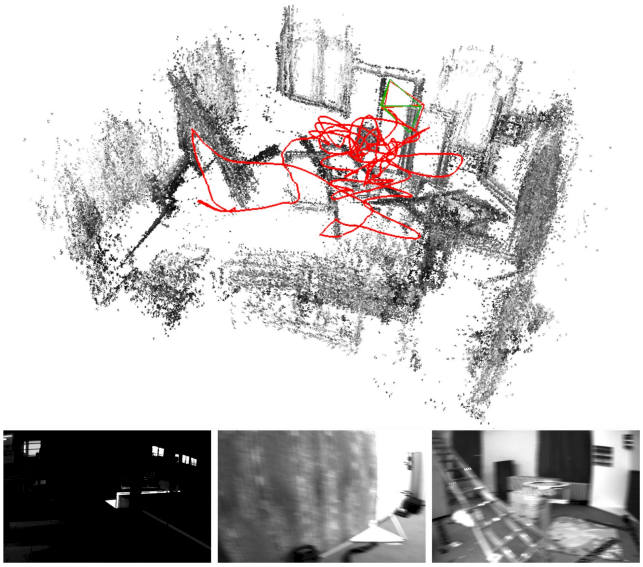


Fig. 1: Bottom: Example images from the EuRoC-dataset: Low illumination, strong motion blur and little texture impose significant challenges for odometry estimation. Still our method is able to process all sequences with a rmse of less then 0.23m. Top: Reconstruction, estimated pose (red camera) and groundtruth pose (green camera) at the end of V1_03_difficult.

Quantitative evaluation on the EuRoC dataset [2] demonstrates that we can reliably determine camera motion and sparse 3D structure (in metric units) from a visual-inertial system on a rapidly moving micro aerial vehicle (MAV) despite challenging illumination conditions (Fig. 1).

In summary, our contributions are:

- a direct sparse visual-inertial odometry system.
- a novel initialization strategy where scale and gravity direction are included into the model and jointly optimized after initialization.
- we introduce "dynamic marginalization" as a technique to adaptively employ marginalization strategies even in cases where certain variables undergo drastic changes.
- an extensive evaluation on the challenging EuRoC dataset showing that both, the overall system and the

[1]The authors are with the Computer Vision Group, Computer Science Institute 9, Technische Universität München, 85748 Garching, Germany {stumberg, usenko, cremers}@in.tum.de

initialization strategy outperform the state of the art.

II. Related work

Motion estimation using cameras and IMUs has been a popular research topic for many years. In this section we will give a summary of visual, and visual-inertial odometry methods. We will also discuss approaches to the initialization of monocular visual-inertial odometry, where the initial orientation, velocity and scale are not known in advance.

The term visual odometry was introduced in the work of Nister et al. [24], who proposed to use frame-to-frame matching of the sparse set of points to estimate the motion of the cameras. Most of the early approaches were based on matching features detected in the images, in particular MonoSLAM [5], a real-time capable EKF-based method. Another prominent example is PTAM [15], which combines a bundle-adjustment backend for mapping with real-time capable tracking of the camera relative to the constructed map. Recently, a feature-based system capable of large-scale real-time SLAM was presented by Mur-Artal et al. [21].

Unlike feature-based methods, direct methods use unprocessed intensities in the image to estimate the motion of the camera. The first real-time capable direct approach for stereo cameras was presented in [4]. Several methods for motion estimation for RGB-D cameras were developed by Kerl et al. [14]. More recently, direct approaches were also applied to monocular cameras, in a dense [23], semi-dense [7], and sparse fashion [10] [6].

Due to the complementary nature of the IMU sensors, there were many attempts to combine them with vision. They provide good short-term motion prediction and make roll and pitch angles observable. At first, vision systems were used just as a provider of 6D pose measurements which were then inserted in the combined optimization. This, so-called *loosely coupled* approach, was presented in [20] and [8]. It is generally easier to implement, since the vision algorithm requires no modifications. On the other hand, *tightly coupled* approaches jointly optimize motion parameters in a combined energy function. They are able to capture more correlations in the multisensory data stream leading to more precision and robustness. Several prominent examples are filtering based approaches [17] [1] and energy-minimization based approaches [16] [9] [26] [22].

Another issue relevant for the practical use of monocular visual-inertial odometry is initialization. Right after the start, the system has no prior information about the initial pose, velocities and depth values of observed points in the image. Since the energy functional that is being minimized is highly non-convex, a bad initialization might result in divergence of the system. The problem is even more complicated, since some types of motion do not allow to uniquely determine all these values. A closed form solution for initialization, together with analysis of the exceptional cases was presented in [19], and extended to consider IMU biases in [12].

III. Direct Sparse Visual-Inertial Odometry

The following approach is based on iterative minimization of photometric and inertial errors in a non-linear optimization framework. To make the problem computationally feasible the optimization is performed on a window of recent frames while all older frames get marginalized out. Our approach is based on [6] and can be viewed as a direct formulation of [16]. In contrast to [26], we jointly determine poses and 3D geometry from a single optimization function. This results in better precision especially on hard sequences. Compared to [9] we perform a full bundle-adjustment like optimization instead of including structure-less vision error terms.

The proposed approach estimates poses and depths by minimizing the energy function

$$E_{\text{total}} = \lambda \cdot E_{\text{photo}} + E_{\text{inertial}} \tag{1}$$

which consists of the photometric error E_{photo} (section III-B) and an inertial error term E_{inertial} (section III-C).

The system contains two main parts running in parallel:

- The coarse tracking is executed for every frame and uses direct image alignment combined with an inertial error term to estimate the pose of the most recent frame.
- When a new keyframe is created we perform a visual-inertial bundle adjustment like optimization that estimates the geometry and poses of all active keyframes.

In contrast to [22] we do not wait for a fixed amount of time before initializing the visual-inertial system but instead we jointly optimize all parameters including the scale. This yields a higher robustness as inertial measurements are used right from the beginning.

A. Notation

Throughout the paper we will use the following notation: bold upper case letters $\mathbf{H}$ represent matrices, bold lower case $\boldsymbol{x}$ vectors and light lower case λ represent scalars. Transformations between coordinate frames are denoted as $\mathbf{T}_{i\_j} \in \mathbf{SE}(3)$ where point in coordinate frame i can be transformed to the coordinate frame j using the following equation $\mathbf{p}_i = \mathbf{T}_{i\_j}\mathbf{p}_j$. We denote Lie algebra elements as $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$, where $\boldsymbol{\xi} \in \mathbb{R}^6$, and use them to apply small increments to the 6D pose $\boldsymbol{\xi}'_{i\_j} = \boldsymbol{\xi}_{i\_j} \boxplus \boldsymbol{\xi} := \log\left(e^{\hat{\boldsymbol{\xi}}_{i\_j}} \cdot e^{\hat{\boldsymbol{\xi}}}\right)^{\vee}$.

We define the *world* as a fixed inertial coordinate frame with gravity acting in negative Z axis. We also assume that the transformation from camera to IMU frame $T_{\text{imu\_cam}}$ is fixed and calibrated in advance. Factor graphs are expressed as a set G of factors and we use $G_1 \cup G_2$ to denote a factor graph containing all factors that are either in G_1 or in G_2.

B. Photometric Error

The photometric error of a point $p \in \Omega_i$ in reference frame i observed in another frame j is defined as follows:

$$E_{\boldsymbol{p}j} = \sum_{\mathbf{p} \in \mathcal{N}_{\boldsymbol{p}}} \omega_{\boldsymbol{p}} \left\| (I_j[\boldsymbol{p'}] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right\|_{\gamma}, \tag{2}$$

where $\mathcal{N}_{\boldsymbol{p}}$ is a small set of pixels around the point $\boldsymbol{p}$, I_i and I_j are images of respective frames, t_i, t_j are the exposure times, a_i, b_i, a_j, b_j are the coefficients to correct for affine illumination changes, γ is the Huber norm, ω_p is a gradient-dependent weighting and $\boldsymbol{p'}$ is the point projected into I_j.

With that we can formulate the photometric error as

$$E_{\text{photo}} = \sum_{i \in \mathcal{F}} \sum_{\boldsymbol{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\boldsymbol{p})} E_{\boldsymbol{p}j}, \tag{3}$$

where $\mathcal{F}$ is a set of keyframes that we are optimizing, $\mathcal{P}_i$ is a sparse set of points in keyframe i, and $obs(\mathbf{p})$ is a set of observations of the same point in other keyframes.

C. Inertial Error

In order to construct the error term that depends on rotational velocities measured by the gyroscope and linear acceleration measured by the accelerometer we use the nonlinear dynamic model defined in [26, eq. (6), (7), (8)].

As IMU data is obtained with a much higher frequency than images we follow the preintegration approach proposed in [18] and improved in [3] and [9]. This allows us to add a single IMU factor describing the pose between two camera frames. For two states $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ (based on the state definition in Equation (9)), and IMU-measurements $\boldsymbol{a}_{i,j}$ and $\boldsymbol{\omega}_{i,j}$ between the two images we obtain a prediction $\widehat{\boldsymbol{s}}_j$ as well as an associated covariance matrix $\widehat{\Sigma}_{s,j}$. The corresponding error function is

$$E_{\text{inertial}}(\boldsymbol{s}_i, \boldsymbol{s}_j) := (\boldsymbol{s}_j \boxminus \widehat{\boldsymbol{s}}_j)^T \, \widehat{\boldsymbol{\Sigma}}_{s,j}^{-1} \, (\boldsymbol{s}_j \boxminus \widehat{\boldsymbol{s}}_j) \tag{4}$$

where the operator $\boxminus$ applies $\boldsymbol{\xi}_j \boxplus \left(\widehat{\boldsymbol{\xi}_j}\right)^{-1}$ for poses and a normal subtraction for other components.

D. IMU Initialization and the problem of observability

In contrast to a purely monocular system the usage of inertial data enables us to observe metric scale and gravity direction. This also implies that those values have to be properly initialized, otherwise optimization might diverge. Initialization of the monocular visual-inertial system is a well studied problem with an excellent summary provided in [19]. [19, Tables I and II] show that for certain motions immediate initialization is not possible, for example when moving with zero acceleration and constant non-zero velocity. To demonstrate that it is a real-world problem and not just a theoretical case we note that the state-of-the-art visual-inertial SLAM system [22] uses the first 15 seconds of camera motion for the initialization on the EuRoC dataset to make sure that all values are observable.

Therefore we propose a novel strategy for handling this issue. We explicitly include scale (and gravity direction) as a parameter in our visual-inertial system and jointly optimize them together with the other values such as poses and geometry. This means that we can initialize with an arbitrary scale instead of waiting until it is observable. We initialize the various parameters as follows.

- We use the same visual initializer as [6] which computes a rough pose estimate between two frames as well as approximate depths for several points. They are normalized so that the average depth is 1.
- The initial gravity direction is computed by averaging up to 40 accelerometer measurements, yielding a sufficiently good estimate even in cases of high acceleration.
- We initialize the velocity and IMU-biases with zero and the scale with 1.0.

All these parameters are then jointly optimized during a bundle adjustment like optimization.

E. $\mathbf{SIM}(3)$-based Representation of the World

In order to be able to start tracking and mapping with a preliminary scale and gravity direction we need to include them into our model. Therefore in addition to the metric coordinate frame we define the DSO coordinate frame to be a scaled and rotated version of it. The transformation from the DSO frame to the metric frame is defined as $\mathbf{T}_{m\_d} \in \{\mathbf{T} \in \mathbf{SIM}(3) \mid \text{translation}(\mathbf{T}) = 0\}$, together with the corresponding $\boldsymbol{\xi}_{m\_d} = \log(\mathbf{T}_{m\_d}) \in \mathfrak{sim}(3)$. We add a superscript D or M to all poses denoting in which coordinate frame they are expressed. In the optimization the photometric error is always evaluated in the DSO frame, making it independent of the scale and gravity direction, whereas the inertial error has to use the metric frame.

F. Scale-aware Visual-inertial Optimization

We optimize the poses, IMU-biases and velocities of a fixed number of keyframes. Fig. 2a shows a factor graph of the problem. Note that there are in fact many separate visual factors connecting two keyframes each, which we have combined to one big factor connecting all the keyframes in this visualization. Each IMU-factor connects two subsequent keyframes using the preintegration scheme described in section III-C. As the error of the preintegration increases with the time between the keyframes we ensure that the time between two consecutive keyframes is not bigger than 0.5 seconds which is similar to what [22] have done. Note that in contrast to their method however we allow the marginalization procedure described in section III-F.2 to violate this constraint which ensures that long-term relationships between keyframes can be properly observed.

An important property of our algorithm is that the optimized poses are not represented in the metric frame but in the DSO frame. This means that they do not depend on the scale of the environment.

1) Nonlinear Optimization: We perform nonlinear optimization using the Gauss-Newton algorithm. For each active keyframe we define a state vector

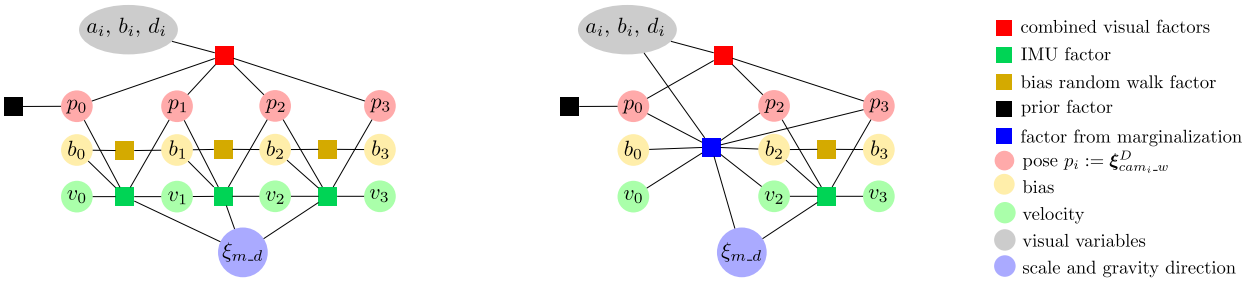
$$\boldsymbol{s}_i := [\left(\boldsymbol{\xi}^D_{cam_i\_w}\right)^T, \boldsymbol{v}_i^T, \boldsymbol{b}_i^T, a_i, b_i, d_i^1, d_i^2, ..., d_i^m]^T \tag{5}$$

where $\boldsymbol{v}_i \in \mathbb{R}^3$ is the velocity, $\boldsymbol{b}_i \in \mathbb{R}^6$ is the current IMU bias, a_i and b_i are the affine illumination parameters used in equation (2) and d_i^j are the inverse depths of the points hosted in this keyframe.

The full state vector is then defined as

$$\boldsymbol{s} = [\boldsymbol{c}^T, \boldsymbol{\xi}_{m\_d}^T, \boldsymbol{s}_1^T, \boldsymbol{s}_2^T, ..., \boldsymbol{s}_n^T]^T \tag{6}$$

where $\boldsymbol{c}$ contains the geometric camera parameters and $\boldsymbol{\xi}_{m\_d}$ denotes the translation-free transformation between the DSO frame and the metric frame as defined in section III-E. We define the operator $\boldsymbol{s} \boxplus \boldsymbol{s}'$ to work on state vectors by



(a) Factor graph for the visual-inertial optimization. (b) Factor graph after keyframe 1 was marginalized.

Fig. 2: Factor graphs for the visual-inertial joint optimization before and after the marginalization of a keyframe.

applying the concatenation operation $\boldsymbol{\xi} \boxplus \boldsymbol{\xi'}$ for Lie algebra components and a plain addition for other components.

Using the stacked residual vector $\boldsymbol{r}$ we define

$$\mathbf{J} = \left.\frac{d\boldsymbol{r}\left(\boldsymbol{s} \boxplus \boldsymbol{\epsilon}\right)}{d\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}, \mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \boldsymbol{b} = -\mathbf{J}^T\mathbf{W}\boldsymbol{r} \tag{7}$$

where $\mathbf{W}$ is a diagonal weight matrix. Then the update that we compute is $\boldsymbol{\delta} = \mathbf{H}^{-1}\boldsymbol{b}$.

Note that the visual energy term E_{photo} and the inertial error term E_{imu} do not have common residuals. Therefore we can divide $\mathbf{H}$ and $\boldsymbol{b}$ each into two independent parts

$$\mathbf{H} = \mathbf{H}_{\text{photo}} + \mathbf{H}_{\text{imu}} \text{ and } \boldsymbol{b} = \boldsymbol{b}_{\text{photo}} + \boldsymbol{b}_{\text{imu}} \tag{8}$$

As the inertial residuals compare the current relative pose to the estimate from the inertial data they need to use poses in the metric frame relative to the IMU. Therefore we define additional state vectors for the inertial residuals.

$$\boldsymbol{s}'_i := [\boldsymbol{\xi}^M_{w\_imu_i}, \boldsymbol{v}_i, \boldsymbol{b}_i]^T \text{ and } \boldsymbol{s}' = \left[\boldsymbol{s}'^T_1, \boldsymbol{s}'^T_2, ..., \boldsymbol{s}'^T_n\right]^T \tag{9}$$

The inertial residuals lead to

$$\mathbf{H}'_{\text{imu}} = \mathbf{J}'^T_{\text{imu}}\mathbf{W}_{\text{imu}}\mathbf{J}'_{\text{imu}} \text{ and } \boldsymbol{b}'_{\text{imu}} = -\mathbf{J}'^T_{\text{imu}}\mathbf{W}_{\text{imu}}\boldsymbol{r}_{\text{imu}} \tag{10}$$

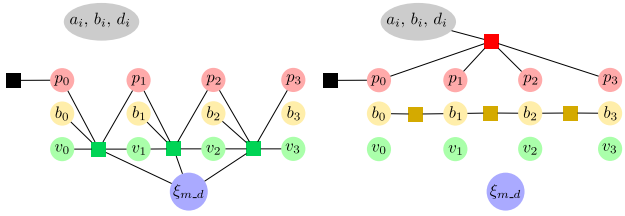
For the joint optimization however we need to obtain $\mathbf{H}_{\text{imu}}$ and $\boldsymbol{b}_{\text{imu}}$ based on the state definition in Equation (6). As the two definitions mainly differ in their representation of the poses we can compute $\mathbf{J}_{\text{rel}}$ such that

$$\mathbf{H}_{\text{imu}} = \mathbf{J}^T_{\text{rel}} \cdot \mathbf{H}'_{\text{imu}} \cdot \mathbf{J}_{\text{rel}} \text{ and } \boldsymbol{b}_{\text{imu}} = \mathbf{J}^T_{\text{rel}} \cdot \boldsymbol{b}'_{\text{imu}} \tag{11}$$

The computation of $\mathbf{J}_{\text{rel}}$ is detailed in the supplementary material. Note that we represent all transformations as elements of $\mathfrak{sim}(3)$ and fix the scale to 1 for all of them except $\boldsymbol{\xi}_{m\_d}$.

2) Marginalization using the Schur-Complement: In order to compute Gauss-Newton updates in a reasonable time-frame we perform partial marginalization for older keyframes. This means that all variables corresponding to this keyframe (pose, bias, velocity and affine illumination parameters) are marginalized out using the Schur complement. Fig. 2b shows how marginalization changes the factor graph.

The marginalization of the visual factors is handled as in [6] by dropping residual terms that affect the sparsity of the system and by first marginalizing all points in the keyframe before marginalizing the keyframe itself.



(a) G_{metric} (b) G_{visual}

Fig. 3: Partitioning of the factor graph from Fig. 2a into G_{metric} and G_{visual}. G_{metric} contains all IMU-factors while G_{visual} contains the factors that do not depend on $\boldsymbol{\xi}_{m\_d}$. Note that both of them do not contain any marginalization factors.

Marginalization is performed using the Schur-complement [6, eq. (16), (17) and (18)]. As the factor resulting from marginalization requires the linearization point of all connected variables to remain fixed we apply [6, eq. (15)] to approximate the energy around further linearization points.

In order to maintain consistency of the system it is important that Jacobians are all evaluated at the same value for variables that are connected to a marginalization factor as otherwise the nullspaces get eliminated. Therefore we apply "First Estimates Jacobians". For the visual factors we follow [6] and evaluate $\mathbf{J}_{\text{photo}}$ and $\mathbf{J}_{\text{geo}}$ at the linearization point. When computing the inertial factors we fix the evaluation point of $\mathbf{J}_{\text{rel}}$ for all variables which are connected to a marginalization factor. Note that this always includes $\boldsymbol{\xi}_{m\_d}$.

3) Dynamic Marginalization for Delayed Scale Convergence: The marginalization procedure described in subsection III-F.2 has two purposes: reduce the computation complexity of the optimization by removing old states and maintain the information about the previous states of the system. This procedure fixes the linearization points of the states connected to the old states, so they should already have a good estimate. In our scenario this is the case for all variables except of scale.

The main idea of "Dynamic marginalization" is to maintain several marginalization priors at the same time and reset the one we currently use when the scale estimate moves too far from the linearization point in the marginalization prior.

In our implementation we use three marginalization priors: M_{visual}, M_{curr} and M_{half}. M_{visual} contains only scale inde-

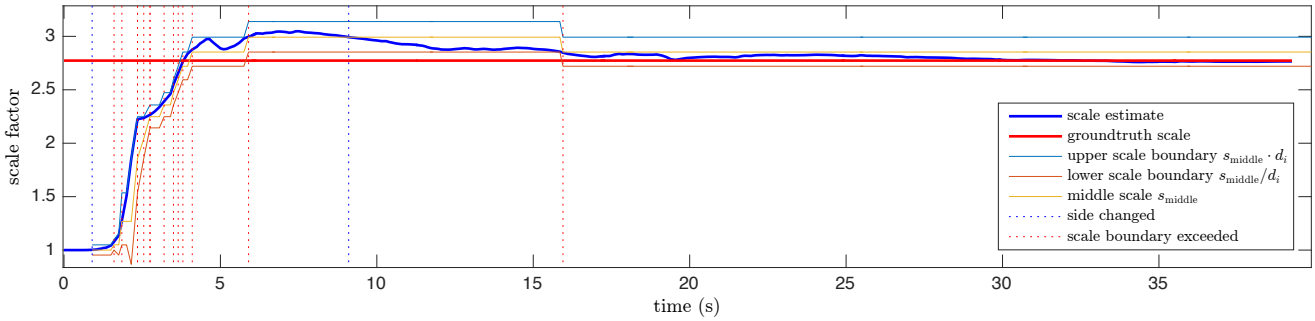


Fig. 4: The scale estimation running on the V1_03_difficult sequence from the EuRoC dataset. We show the current scale estimate (bold blue), the groundtruth scale (bold red) and the current scale interval (light lines). The vertical dotted lines denote when the side changes (blue) and when the boundary of the scale interval is exceeded (red). In practice this means that M_{curr} contains the inertial factors since the last blue or red dotted line that is before the last red dotted line. For example at 16s it contains all inertial data since the blue line at 9 seconds.

pendent information from previous states of the vision and cannot be used to infer the global scale. M_{curr} contains all information since the time we set the linearization point for the scale and M_{half} contains only the recent states that have a scale close to the current estimate.

When the scale estimate deviates too much from the linearization point of M_{curr}, the value of M_{curr} is set to M_{half} and M_{half} is set to M_{visual} with corresponding changes in the linearization points. This ensures that the optimization always has some information about the previous states with consistent scale estimates. In the remaining part of the section we provide the details of our implementation.

We define G_{metric} to contain only the visual-inertial factors (which depend on $\boldsymbol{\xi}_{m\_d}$) and G_{visual} to contain all other factors, except the marginalization priors. Then

$$G_{\text{full}} = G_{\text{metric}} \cup G_{\text{visual}} \tag{12}$$

Fig. 3 depicts the partitioning of the factor graph.

We define three different marginalization factors M_{curr}, M_{visual} and M_{half}. For the optimization we always compute updates using the graph

$$G_{ba} = G_{\text{metric}} \cup G_{\text{visual}} \cup M_{\text{curr}} \tag{13}$$

When keyframe i is marginalized we update M_{visual} with the factor arising from marginalizing frame i in $G_{\text{visual}} \cup M_{\text{visual}}$. This means that M_{visual} contains all marginalized visual factors and no marginalized inertial factors making it independent of the scale.

For each marginalized keyframe i we define

$$s_i := \text{scale estimate at the time, } i \text{ was marginalized} \tag{14}$$

We define $i \in M$ if and only if M contains an *inertial* factor that was marginalized at time i. Using this we enforce the following constraints for inertial factors.

$$\forall i \in M_{\text{curr}} : s_i \in [s_{\text{middle}}/d_i, s_{\text{middle}} \cdot d_i] \tag{15}$$

$$\forall i \in M_{\text{half}} : s_i \in \begin{cases} [s_{\text{middle}}, s_{\text{middle}} \cdot d_i], & \text{if } s_{\text{curr}} > s_{\text{middle}} \\ [s_{\text{middle}}/d_i, s_{\text{middle}}], & \text{otherwise} \end{cases} \tag{16}$$

where s_{middle} is the current middle of the allowed scale interval (initialized with s_0), d_i is the size of the scale interval at time i, and s_{curr} is the current scale estimate.

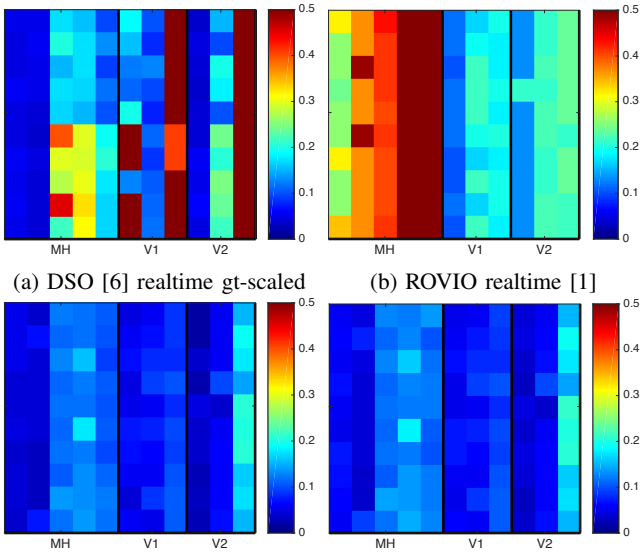
We update M_{curr} by marginalizing frame i in G_{ba} and we update M_{half} by marginalizing i in $G_{\text{metric}} \cup G_{\text{visual}} \cup M_{\text{half}}$

In order to preserve the constraints in Equations (15) and (16) we apply Algorithm 1 everytime a marginalization happens. By following these steps on the one hand we make sure that the constraints are satisfied which ensures that the scale difference in the currently used marginalization factor stays smaller than d_i^2. On the other hand the factor always contains some inertial factors so that the scale estimation works at all times. Note also that M_{curr} and M_{half} have separate First Estimate Jacobians that are employed when the respective marginalization factor is used. Fig. 4 shows how the system works in practice.

Algorithm 1 Constrain Marginalization

upper $\leftarrow s_{\text{curr}} > s_{\text{middle}}$
if upper $\neq$ lastUpper **then** ▷ Side changes.
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
end if
if $s_{\text{curr}} > s_{\text{middle}} \cdot d_i$ **then** ▷ Upper boundary exceeded.
  $M_{\text{curr}} \leftarrow M_{\text{half}}$
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
  $s_{\text{middle}} \leftarrow s_{\text{middle}} \cdot d_i$
end if
if $s_{\text{curr}} < s_{\text{middle}}/d_i$ **then** ▷ Lower boundary exceeded.
  $M_{\text{curr}} \leftarrow M_{\text{half}}$
  $M_{\text{half}} \leftarrow M_{\text{visual}}$
  $s_{\text{middle}} \leftarrow s_{\text{middle}}/d_i$
end if
lastUpper $\leftarrow$ upper

An important part of this strategy is the choice of d_i. It should be small, in order to keep the system consistent, but not too small so that M_{curr} always contains enough inertial factors. Therefore we chose to dynamically adjust



(a) DSO [6] realtime gt-scaled (b) ROVIO realtime [1]

(c) our method realtime gt-scaled (d) our method realtime

Fig. 5: rmse for different methods run 10 times (lines) on each sequence (columns) of the EuRoC dataset.

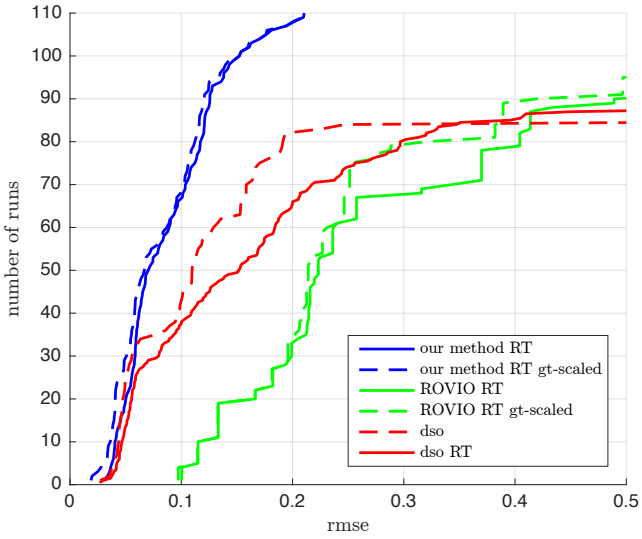


Fig. 6: Cumulative error plot on the EuRoC-dataset (RT means realtime). This experiment demonstrates that the additional IMU not only provides a reliable scale estimate, but that it also significantly increases accuracy and robustness.

the parameter as follows. At all time steps i we calculate

$$d_i = \min \{d_{\text{min}}^j \mid j \in \mathbb{N} \setminus \{0\}, \frac{s_i}{s_{i-1}} < d_i\} \qquad (17)$$

This ensures that it cannot happen that the M_{half} gets reset to M_{visual} at the same time that M_{curr} is exchanged with M_{half}. Therefore it prevents situations where M_{curr} contains no inertial factors at all, making the scale estimation more reliable. In our experiments we chose $d_{\text{min}} = \sqrt{1.1}$.

G. Coarse Visual-Inertial Tracking

The coarse tracking is responsible for computing a fast pose estimate for each frame that also serves as an initialization for the joint optimization detailed in III-F. We perform conventional direct image alignment between the current frame and the latest keyframe, while keeping the geometry and the scale fixed. Inertial residuals using the previously described IMU preintegration scheme are placed between subsequent frames. Everytime the joint optimization is finished for a new frame, the coarse tracking is reinitialized with the new estimates for scale, gravity direction, bias, and velocity as well as the new keyframe as a reference for the visual factors. Similar to the joint optimization we perform partial marginalization to keep the update time constrained. After estimating the variables for a new frame we marginalize out all variables except the keyframe pose and the variables of the newest frame. In contrast to the joint optimization we do not need to use dynamic marginalization because the scale is not included in the optimization.

IV. Results

We evaluate our approach on the publicly available EuRoC dataset [2]. The performance is compared to [6], [1], [21], [26], [16] and [13]. We also provide supplementary material with more evaluation and a video at vision.in.tum.de/vi-dso.

A. Robust Quantitative Evaluation

In order to obtain an accurate evaluation we run our method 10 times for each sequence of the dataset (using the left camera). We directly compare the results to visual-only DSO [6] and ROVIO [1]. As DSO cannot observe the scale we evaluate using the optimal ground truth scale in some plots (with the description "gt-scaled") to enable a fair comparison. For all other results we scale the trajectory with the final scale estimate (our method) or with 1 (other methods). For DSO we use the results published together with their paper. We use the same start and end times for each sequence to run our method and ROVIO. Note that the drone has a high initial velocity in some sequences when using these start times making it especially challenging for our IMU initialization. Fig. 5 shows the root mean square error (rmse) for every run and Fig. 6 displays the cumulative error plot. Clearly our method significantly outperforms DSO and ROVIO. Without inertial data DSO is not able to work on all sequences especially on V1_03_difficult and V2_03_difficult and it is also not able to scale the results correctly. ROVIO on the other hand is very robust but as a filtering-based method it cannot provide sufficient accuracy.

Table I shows a comparison to several other methods. For our results we have displayed the median error for each sequence from the 10 runs plotted in Fig. 5c. This makes the results very meaningful. For the other methods unfortunately only one result was reported so we have to assume that they are representative as well. The results for [16] and [13] were taken from [13]. The results for [21] (as reported in their paper) differ slightly from the other methods as they show the error of the keyframe trajectory instead of the full trajectory. This is a slight advantage as keyframes are bundle-adjusted in their method which does not happen for the other frames.

TABLE I: Accuracy of the estimated trajectory on the EuRoC dataset for several methods. Note that ORB-SLAM does a convincing job showing leading performance on some of the sequences. Nevertheless, since our method directly works on the sensor data (colors and IMU measurements), we observe similar precision and a better robustness – even without loop closuring. Moreover, the proposed method is the only one not to fail on any of the sequences.

Sequence		MH1	MH2	MH3	MH4	MH5	V11	V12	V13	V21	V22	V23
VI-DSO (our method, RT) (median of 10 runs each)	RMSE	**0.062**	**0.044**	0.117	**0.132**	0.121	0.059	0.067	**0.096**	0.040	0.062	0.174
	RMSE gt-scaled	0.041	0.041	0.116	0.129	0.106	0.057	0.066	0.095	0.031	0.060	0.173
	Scale Error (%)	1.1	**0.5**	**0.4**	**0.2**	0.8	1.1	1.1	**0.8**	1.2	**0.3**	**0.4**
VI ORB-SLAM (keyframe trajectory)	RMSE	0.075	0.084	**0.087**	0.217	**0.082**	**0.027**	**0.028**	X	**0.032**	**0.041**	**0.074**
	RMSE gt-scaled	0.072	0.078	0.067	0.081	0.077	0.019	0.024	X	0.031	0.026	0.073
	Scale Error (%)	**0.5**	0.8	1.5	3.4	**0.5**	**0.9**	**0.8**	X	**0.2**	1.4	0.7
VI odometry [16], mono	RMSE	0.34	0.36	0.30	0.48	0.47	0.12	0.16	0.24	0.12	0.22	X
VI odometry [16], stereo	RMSE	0.23	0.15	0.23	0.32	0.36	0.04	0.08	0.13	0.10	0.17	X
VI SLAM [13], mono	RMSE	0.25	0.18	0.21	0.30	0.35	0.11	0.13	0.20	0.12	0.20	X
VI SLAM [13], stereo	RMSE	0.11	0.09	0.19	0.27	0.23	0.04	0.05	0.11	0.10	0.18	X

In comparison to VI ORB-SLAM our method outperforms it in terms of rmse on several sequences. As ORB-SLAM is a SLAM system while ours is a pure odometry method this is a remarkable achievement especially considering the differences in the evaluation. Note that the Vicon room sequences (V*) are executed in a small room and contain a lot of loopy motions where the loop closures done by a SLAM system significantly improve the performance. Also our method is more robust as ORB-SLAM fails to track one sequence. Even considering only sequences where ORB-SLAM works our approach has a lower maximum rmse.

Compared to [16] and [13] our method obviously outperforms them. It is better than the monocular versions on every single sequence and it beats even the stereo and SLAM-versions on 9 out of 11 sequences.

In summary our method is the only one which is able to track all the sequences successfully except ROVIO.

We also compare the Relative Pose Error to [21] and [26] on the V1_0*-sequences of EuRoC (Fig. 7). While our method cannot beat the SLAM system and the stereo method on the easy sequence we outperform [26] and are as good as [21] on the medium sequence. On the hard sequence we outperform both of the contenders even though we neither use stereo nor loop-closures.

B. Evaluation of the Initialization

There are only few methods we can compare our initialization to. Some approaches like [19] have not been tested on real data. While [12] provides results on real data, the dataset used was featuring a downward-looking camera and an environment with a lot of features which is not comparable to the EuRoC-dataset in terms of difficulty. Also they do not address the problem of late observability which suggests that a proper motion is performed in the beginning of their dataset. As a filtering-based method ROVIO does not need a specific initialization procedure but it also cannot compete in terms of accuracy making it less relevant for this discussion. Visual-inertial LSD-SLAM uses stereo and therefore does not face the main problem of scale estimation. Therefore we compare our initialization procedure to visual-inertial ORB-SLAM [21] as both of the methods work on the challenging EuRoC-dataset and have to estimate the scale, gravity direction, bias, and velocity.

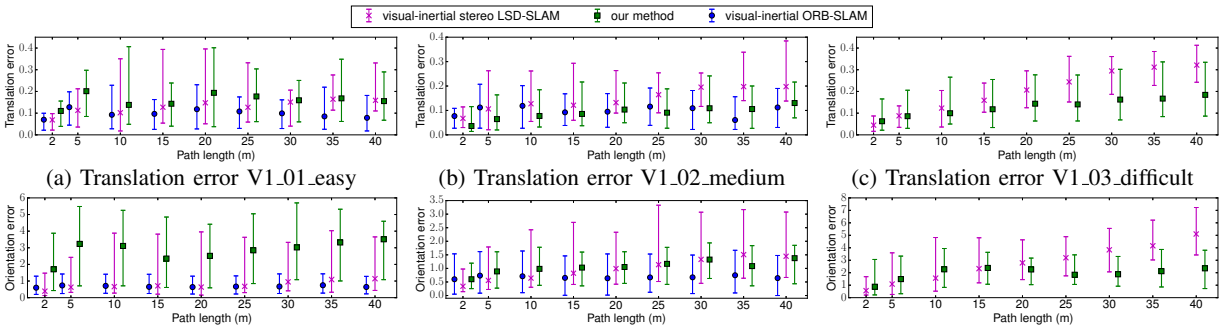
In comparison to [21] our estimated scale is better overall (Table I). On most sequences our method provides a better scale, and our average scale error (0.7% compared to 1.0%) as well as our maximum scale error (1.2% compared to 3.4%) is lower. In addition our method is more robust as the initialization procedure of [21] fails on V1_03_difficult.

Apart from the numbers we argue that our approach is superior in terms of the general structure. While [21] have to wait for 15 seconds until the initialization is performed, our method provides an approximate scale and gravity direction almost instantly, that gets enhanced over time. Whereas in [21] the pose estimation has to work for 15 seconds without any IMU data, in our method the inertial data is used to improve the pose estimation from the beginning. This is probably one of the reasons why our method is able to process V1_03_difficult. Finally our method is better suited for robotics applications. For example an autonomous drone is not able to fly without gravity direction and scale for 15 seconds and hope that afterwards the scale was observable. In contrast our method offers both of them right from the start. The continuous rescaling is also not a big problem as an application could use the unscaled measurements for building a consistent map and for providing flight goals, whereas the scaled measurements can be used for the controller. Fig. 8 shows the scale estimation for MH_04.

Overall we argue that our initialization procedure exceeds the state of the art and think that the concept of initialization with a very rough scale estimate and jointly estimating it during pose estimation will be a useful concept in the future.

V. Conclusion

We have presented a novel formulation of direct sparse visual-inertial odometry. We explicitely include scale and gravity direction in our model in order to deal with cases where the scale is not immediately observable. As the initial



(a) Translation error V1_01_easy (b) Translation error V1_02_medium (c) Translation error V1_03_difficult

(d) Orientation error V1_01_easy (e) Orientation error V1_02_medium (f) Orientation error V1_03_difficult

Fig. 7: Relative Pose Error evaluated on three sequences of the EuRoC-dataset for visual-inertial ORB-SLAM [21], visual-inertial stereo LSD-SLAM [26] and our method. Although the proposed VI-DSO does not use loop closuring (like [21]) or stereo (like [26]), VI-DSO is quite competitive in terms of accuracy and robustness. Note that [21] with loop closures is slightly more accurate on average, yet it entirely failed on V1_03_difficult.

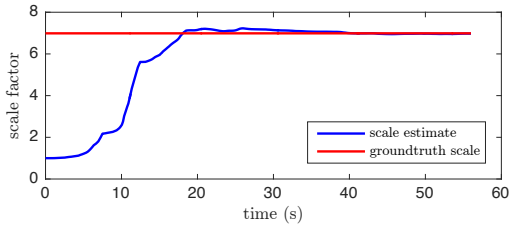


Fig. 8: Scale estimate for MH_04_difficult (median result of 10 runs in terms of tracking accuracy). Note how the estimated scale converges to the correct value despite being initialized far from the optimum.

scale can be very far from the optimum we have proposed a novel technique called dynamic marginalization where we maintain multiple marginalization priors and constrain the maximum scale difference. Extensive quantitative evaluation demonstrates that the proposed visual-inertial odometry method outperforms the state of the art, both the complete system as well as the IMU initialization procedure. In particular, experiments confirm that the inertial information not only provides a reliable scale estimate, but it also drastically increases precision and robustness.

Acknowledgements

We thank Jakob Engel for releasing the code of DSO and for his helpful comments on First Estimates Jacobians, and the authors of [21] for providing their numbers for the comparison in Fig. 7.

References

[1] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IEEE/RSJ IROS*, 2015.

[2] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *IJRR*, 2016.

[3] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *IEEE ICRA*, 2014.

[4] A. Comport, E. Malis, and P. Rives, "Accurate quadri-focal tracking for robust 3D visual odometry," in *IEEE ICRA*, 2007.

[5] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *TPAMI*, vol. 29, 2007.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *TPAMI*, vol. 40, 2018.

[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. of ECCV*, 2014.

[8] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *IEEE/RSJ IROS*, 2012.

[9] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of RSS*, 2015.

[10] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *IEEE ICRA*, 2014.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE CVPR*, 2012.

[12] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robot. and Autom. Lett.*, vol. 2, no. 1, 2017.

[13] A. Kasyanov, F. Engelmann, J. Stückler, and B. Leibe, "Keyframe-Based Visual-Inertial Online SLAM with Relocalization," *ArXiv e-prints:1702.02175*, 2017.

[14] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE ICRA*, 2013.

[15] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. of ISMAR*, 2007.

[16] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *IJRR*, 2014.

[17] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *IJRR*, vol. 32, 2013.

[18] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[19] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *IJCV*, vol. 106, no. 2, 2014.

[20] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *IEEE ICRA*, 2011.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Trans. on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[22] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robot. and Autom. Lett.*, vol. 2, no. 2, 2017.

[23] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE ICCV*, 2011.

[24] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE CVPR*, vol. 1, June 2004, pp. I–652–I–659 Vol.1.

[25] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *IJRR*, 2012.

[26] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *IEEE ICRA*, May 2016.

DM-VIO: Delayed Marginalization Visual-Inertial Odometry

Lukas von Stumberg[1] and Daniel Cremers[1]

***Abstract*— We present DM-VIO, a monocular visual-inertial odometry system based on two novel techniques called delayed marginalization and pose graph bundle adjustment. DM-VIO performs photometric bundle adjustment with a dynamic weight for visual residuals. We adopt marginalization, which is a popular strategy to keep the update time constrained, but it cannot easily be reversed, and linearization points of connected variables have to be fixed. To overcome this we propose delayed marginalization: The idea is to maintain a second factor graph, where marginalization is delayed. This allows us to later readvance this delayed graph, yielding an updated marginalization prior with new and consistent linearization points. In addition, delayed marginalization enables us to inject IMU information into already marginalized states. This is the foundation of the proposed pose graph bundle adjustment, which we use for IMU initialization. In contrast to prior works on IMU initialization, it is able to capture the full photometric uncertainty, improving the scale estimation. In order to cope with initially unobservable scale, we continue to optimize scale and gravity direction in the main system after IMU initialization is complete. We evaluate our system on the EuRoC, TUM-VI, and 4Seasons datasets, which comprise flying drone, large-scale handheld, and automotive scenarios. Thanks to the proposed IMU initialization, our system exceeds the state of the art in visual-inertial odometry, even outperforming stereo-inertial methods while using only a single camera and IMU. The code will be published at `vision.in.tum.de/dm-vio`**

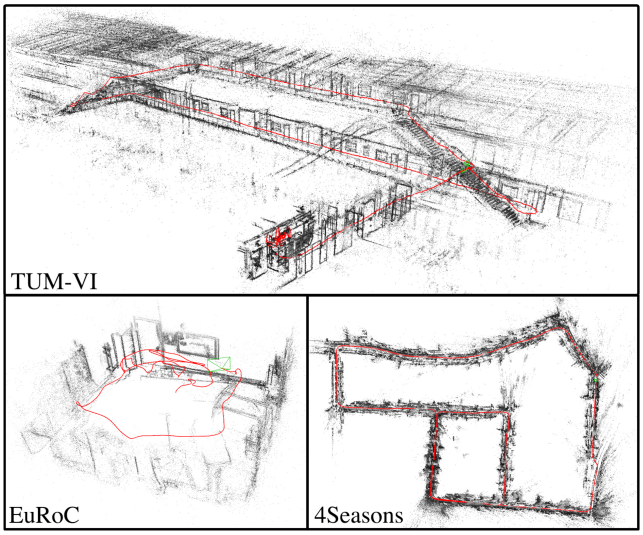


Fig. 1: In this paper, we propose a novel method for monocular visual-inertial odometry. It provides state-of-the-art performance on three different benchmarks. Here we show pointclouds and trajectories (red) for magistrale5, V203_difficult, and neighbor_2020-03-26_13-32-55_0.

I. Introduction

Visual-(inertial) odometry is an increasingly relevant task with applications in robotics, autonomous driving, and augmented reality. A combination of cameras and inertial measurement units (IMUs) for this task is a popular and sensible choice, as they are complementary sensors, resulting in a highly accurate and robust system [1]. In the minimal configuration of a single camera, the IMU can also be used to recover the metric scale. However, the scale is not always observable, the most common degenerate case being movement with a constant velocity [2]. Hence, initialization of such system can take arbitrarily long, depending on the trajectory. Even worse, when initialized prematurely the IMU can in fact worsen the performance. The difficulty of IMU initialization is why stereo-inertial methods have outperformed mono-inertial ones in the past.

Most prior systems [3] [4] [5] initially run visual-only odometry and an IMU initialization in parallel. Once finished, the visual-inertial system is started. This introduces a trade-off for the duration of the initialization period: It should be as short as possible, as no IMU information is used in the main system in the meantime. But when too short, the scale estimate will be inaccurate, leading to bad performance.

VI-DSO [6] instead initializes immediately with an arbitrary scale, and explicitly optimizes the scale in the main system. This yields highly accurate scale estimates, but it can significantly increase the time until the scale is correctly estimated. Also, it can fail in cases where the initial scale error is very high, like in large-scale outdoor environments.

We propose a combination of the two strategies: Similar to the former, we start with a visual-only system and run an IMU initializer in parallel. But after IMU initialization we still estimate scale and gravity direction as explicit optimization variables in the main system. This results in a quickly converging and highly accurate system.

This initialization strategy can lead to three questions: 1) How can the visual uncertainty be properly captured in the IMU initializer. 2) How can information about scale and IMU variables be transferred from the IMU initializer to the main system? 3) If the scale estimate changes, how can a consistent marginalization prior be maintained? VI-DSO [6] tried to address 3 by introducing dynamic marginalization, which does keep the marginalization factor consistent, but loses too much information in the process.

In this work we propose delayed marginalization, which provides a meaningful answer to all three of these questions.

[1]All authors are with the Computer Vision Group, Technical University of Munich, Germany. lukas.stumberg@tum.de, cremers@in.tum.de

The idea is to maintain a second, *delayed* marginalization prior, which has very little overhead, but enables three core techniques:

1) We can populate the delayed factor graph with new IMU factors to perform the proposed pose graph bundle adjustment (PGBA). This is the basis of an IMU initialization which captures the full photometric uncertainty, leading to increased accuracy.
2) The graph used for IMU initialization can be *re-advanced*, providing a marginalization prior with IMU information for the main system.
3) When the scale changes significantly in the main system we can trigger *marginalization replacement*.

The combination of these techniques makes for a highly accurate initializer, which is robust even to long periods of unobservability. Based on it we implement a visual-inertial odometry (VIO) system featuring a photometric front-end integrated with a new dynamic photometric weight.

We evaluate our method on three challenging datasets (Fig. 1), capturing three domains: The EuRoC dataset [7] recorded by a flying drone, the TUM-VI dataset [8] captured with a handheld device, and the 4Seasons dataset [9] representing the automotive scenario. The latter features long stretches of constant velocity, posing a particular challenge for mono-inertial odometry.

We show that our system exceeds the state of the art in visual-inertial odometry, even outperforming stereo-inertial methods. In summary our contributions are:

- Delayed marginalization compensates drawbacks of marginalization while retaining the advantages.
- Pose graph bundle adjustment (PGBA) combines the efficiency of pose graph optimization with the full uncertainty of bundle adjustment.
- A state-of-the-art visual-inertial odometry system with a novel multi-stage IMU initializer and dynamically weighted photometric factors.

The full source code for our approach will be released.

II. Related Work

Initially, most visual odometry and SLAM systems have been feature-based [10], either using filtering [11] or non-linear optimization [12] [13]. More recently, direct methods have been proposed, which optimize a photometric error function and can operate on dense [14] [15], semi-dense [16], or sparse point clouds [17].

Mourikis and Roumeliotis [1] have shown that a tight integration of visual and inertial measurements can greatly increase accuracy and robustness of odometry. Afterwards, many tightly-coupled visual-inertial odometry [18] [19] and SLAM systems [20] [21] [3] [5] have been proposed.

Initialization of monocular visual-inertial systems is not trivial, as sufficient motion is necessary for the scale to become observable [22] [2]. Most systems [4] [3] [5] start with a visual-only system and use its output for a separate IMU initialization. In contrast to these systems, we continue optimizing the scale explicitly in the main system. We note that ORB-SLAM3 [5] also continues to refine the scale after initialization, but this is a separate optimization fixing all poses and only performed until 75 seconds after initialization. [23] also continues to optimize the scale in the main system, but in contrast to us they do not transfer covariances between the main system and the initializer, thus they do not achieve the same level of accuracy. Different to all these systems, the proposed delayed marginalization allows our IMU initializer to capture the full visual uncertainty and continuously optimize the scale in the main system.

VI-DSO [6] initializes immediately with an arbitrary scale and explicitly optimizes the scale in the main system. It also introduced dynamic marginalization to handle the consequential large scale changes in the main system. Compared to it we propose a separate IMU initializer, delayed marginalization as a better alternative to dynamic marginalization, a dynamic photometric error weight, and more improvements, resulting in greatly improved accuracy and robustness.

III. Method

A. Notation

We denote vectors as bold lowercase letters $\mathbf{x}$, matrices as bold upper-case letter $\mathbf{H}$, scalars as lowercase letters λ, and functions as uppercase letters E. $\mathbf{T}^V_{\text{w\_cam}_i} \in \mathbf{SE}(3)$ represents the transformation from camera i to world in the visual coordinate frame V, and $\mathbf{R}^V_{\text{w\_cam}_i} \in \mathbf{SO}(3)$ is the respective rotation. Poses are represented either in visual frame $\mathbf{P}^V_i := \mathbf{T}^V_{\text{cam}_i\text{\_w}}$, or in inertial frame $\mathbf{P}^I_i := \mathbf{T}^I_{\text{w\_imu}_i}$. If not mentioned otherwise we use poses in visual frame $\mathbf{P}_i := \mathbf{P}^V_i$. We also use states $\mathbf{s}$, which can contain transformations, rotations, and vectors. For states we define the subtraction operator $\mathbf{s}_i \boxminus \mathbf{s}_j$, which applies $\log(\mathbf{R}_i \mathbf{R}_j^{-1})$ for rotations and other Lie group elements, and a regular subtraction for vector values.

B. Direct Visual-Inertial Bundle Adjustment

The core of DM-VIO is the visual-inertial bundle adjustment performed for all keyframes. As commonly done, we jointly optimize visual and IMU variables in a combined energy function. For the visual part we choose a direct formulation based on DSO [17], as it is a very accurate and robust system. For integrating IMU data into the bundle adjustment we perform preintegration [24] between keyframes.

We optimize the following energy function using the Levenberg-Marquardt algorithm:

$$E(\mathbf{s}) = W(e_{\text{photo}}) \cdot E_{\text{photo}} + E_{\text{imu}} + E_{\text{prior}} \tag{1}$$

E_{prior} contains added priors on the first pose and the gravity direction, as well as the marginalization priors explained in section III-C. In the following we describe the individual energy terms and the optimized state.

Photometric error: The photometric energy is based on [17]. We optimize a set of active keyframes $\mathcal{F}$, each of which hosts a set of points $\mathcal{P}_i$. Every point $\mathbf{p}$ is projected into all keyframes $\text{obs}(\mathbf{p})$ where it is visible, and the photometric energy is computed:

$$E_{\text{photo}} = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j} \tag{2}$$

$$E_{\mathbf{p}j} = \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} \omega_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma \quad (3)$$

For details regarding the variables we refer the reader to [17].

Dynamic photometric weight: In cases of bad image quality, the system should rely mostly on the inertial data. However due to the photometric cost function used, bad image quality will often lead to very large photometric residuals, effectively increasing the photometric weight compared to the IMU. To counteract this we propose a dynamic photometric weight $W(e_{\text{photo}})$. We compute it using the root mean squared photometric error $e_{\text{photo}} = \sqrt{E_{\text{photo}}/n_{\text{residuals}}}$.

$$W(e_{\text{photo}}) = \lambda \cdot \begin{cases} (\theta/e_{\text{photo}})^2, & \text{if } e_{\text{photo}} \geq \theta \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

where λ is a static weight component, and θ is the threshold from which the error-dependent weight is activated. This effectively normalizes the root mean squared photometric error to be $\sqrt{\lambda}\theta$ at maximum, similar to a threshold robust cost function [25]. In contrast to the Huber norm in Equation (3), which downweights individual points that violate the photometric assumption, this weight addresses cases where the overall image quality is bad and increases the relative weight of the IMU. In our experiments we choose $\theta = 8$.

Optimized variables: We optimize scale and gravity direction as explicit variables. While bundle adjustment can in principle also change the scale and global orientation, convergence is improved when optimizing them explicitly instead [6]. To facilitate this, we represent poses for the visual factors in visual frame V and poses for the IMU factors in IMU frame I. Whereas the IMU frame has a metric scale and a z-axis aligned with gravity direction, the visual frame can have an arbitrary scale and rotation, which is defined during initialization of the visual system. To model this we optimize the scale s and the rotation $\mathbf{R}_{V\_I}$. As yaw is not observable using an IMU, we fix the last coordinate of $\mathbf{R}_{V\_I}$. We convert between the coordinate frames using:

$$\begin{aligned} \mathbf{P}_i^I := \mathbf{T}^I_{\text{w\_imu}_i} = \Omega(\mathbf{P}_i^V, \mathbf{S}, \mathbf{R}_{V\_I}) = \\ \mathbf{R}_{V\_I}^{-1} \mathbf{S}_{I\_V} (\mathbf{P}_i^V)^{-1} \mathbf{S}_{I\_V}^{-1} \mathbf{T}_{\text{cam\_imu}} \end{aligned} \quad (5)$$

where $\mathbf{S}_{I\_V}$ is the $\mathbf{Sim}(3)$ element with identity rotation and translation, and scale s. The other variables are converted to $\mathbf{Sim}(3)$, but note that the result has scale 1 and is in $\mathbf{SE}(3)$.

The full state optimized is

$$\mathbf{s} = \{s, \mathbf{R}_{V\_I}\} \cup \bigcup_{i\in\mathcal{F}} \mathbf{s}_i \quad (6)$$

with $\mathbf{s}_i$ being the states for all active keyframes defined as:

$$\mathbf{s}_i = \{\mathbf{P}_i^V, \mathbf{v}_i, \mathbf{b}_i, a_i, b_i, d_i^0, d_i^2, ...d_i^j\} \quad (7)$$

where $\mathbf{v}_i$ is the velocity, $\mathbf{b}_i$ the bias, a_i and b_i are affine brightness parameters, and d_i^j are the inverse depths of active points hosted in the keyframe. Optimization is performed with a custom integration of the SIMD-accelerated code from [17] for photometric residuals and GTSAM for other factors.

IMU Error: We apply the well-known IMU preintegration first proposed in [26], implemented as smart factors in [27], and further improved in [24]. For this energy we use the IMU state $\mathbf{s}_i^I := \{\mathbf{P}_i^I, \mathbf{v}_i, \mathbf{b}_i\}$, which contains poses in IMU frame and is computed from the optimized state $\mathbf{s}_i$ using Equation (5). Given the previous state $\mathbf{s}_i^I$, the preintegration data provides us with a prediction $\widehat{\mathbf{s}}_j^I$ for the following state $\mathbf{s}_j^I$ as well as a covariance matrix $\widehat{\mathbf{\Sigma}}_j$. The resulting inertial error function penalizes deviations of the current state estimate from the predicted state.

$$E_{\text{imu}}(\mathbf{s}_i^I, \mathbf{s}_j^I) := \left(\widehat{\mathbf{s}}_j^I \boxminus \mathbf{s}_j^I\right)^T \widehat{\mathbf{\Sigma}}_j^{-1} \left(\widehat{\mathbf{s}}_j^I \boxminus \mathbf{s}_j^I\right) \quad (8)$$

C. Partial Marginalization using the Schur Complement

We marginalize old variables using the Schur complement. When marginalizing a set β of variables, we gather all factors dependent on them as well as the connected variables α, which form the Markov blanket. These factors are linearized at the current state estimate, yielding the linear system:

$$\begin{bmatrix} \mathbf{H}_{\alpha\alpha} & \mathbf{H}_{\alpha\beta} \\ \mathbf{H}_{\beta\alpha} & \mathbf{H}_{\beta\beta} \end{bmatrix} \begin{bmatrix} \mathbf{s}_\alpha \\ \mathbf{s}_\beta \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\alpha \\ \mathbf{b}_\beta \end{bmatrix} \quad (9)$$

We apply the Schur-complement, which results in the new linear system $\widehat{\mathbf{H}_{\alpha\alpha}}\boldsymbol{s}_\alpha = \widehat{\boldsymbol{b}_\alpha}$ with

$$\widehat{\mathbf{H}_{\alpha\alpha}} = \mathbf{H}_{\alpha\alpha} - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\mathbf{H}_{\beta\alpha} \quad (10)$$

$$\widehat{\boldsymbol{b}_\alpha} = \boldsymbol{b}_\alpha - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\boldsymbol{b}_\beta \quad (11)$$

This linear system forms a marginalization prior connecting all variables in α (Fig. 2a).

We keep a maximum of $N_f = 8$ keyframes during the bundle adjustment[1]. The marginalization strategy is taken over from [17]: This means that different from a fixed-lag smoother, we do not always marginalize the oldest pose, but instead keep a combination of newer and older poses, as long as they do not leave the field of view. As shown in [17] this is superior to a fixed-lag smoother for visual odometry. When marginalizing a pose, first all remaining points hosted in the frame are marginalized and residuals with remaining active points are dropped. This retains sparsity of the Hessian while preserving enough information.

D. Delayed Marginalization

The concept of marginalization explained in the previous section has the advantage of capturing the full probability distribution. In fact, solving the resulting smaller system is equivalent to solving the much larger original system, as long as the marginalized factors are not relinearized.

However, it also comes with severe drawbacks: Reverting the marginalization of a set of variables is not possible without redoing the whole marginalization procedure. Also, to keep the marginalization prior consistent, First-Estimates Jacobians (FEJ) [28] have to be applied. This means that the linearization point of all connected variables has to be fixed as soon as they are connected to a marginalization prior. This is especially problematic for visual-inertial odometry, where the scale is connected to the marginalization prior as

[1]We define N_f as the maximum number of frames during bundle adjustment, whereas in [17] it is the number of frames after marginalization.

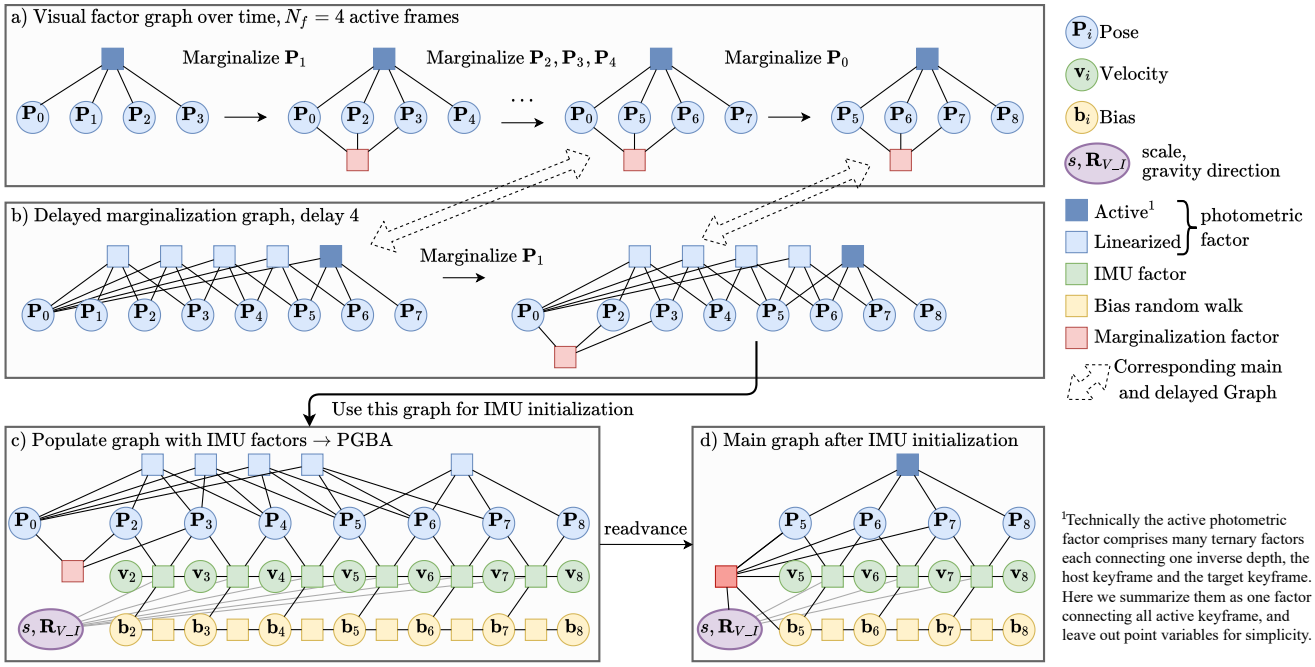


Fig. 2: Delayed Marginalization and PGBA: a) Normal marginalization in the visual graph. Note that not always the oldest pose is marginalized. b) Delayed marginalization: We marginalize all variables in the same order as the main graph, but with a delay d (in practice $d = 100$). Marginalization in this graph is equally fast as marginalizing in the main graph. c) For the pose graph bundle adjustment (PGBA) we populate the delayed graph with IMU factors. This optimization leverages the full photometric uncertainty. d) We readvance the marginalization in the graph used for PGBA to obtain an updated marginalization prior for the main system. This transfers inertial information from the initializer to the main system.

soon as the first keyframe is marginalized, but might change significantly. In [6] dynamic marginalization was introduced to combat this, but it is limited in its application to a single one-dimensional variable, namely the scale, and loses most prior inertial information when the scale changes quickly.

Here we introduce delayed marginalization which circumvents the drawbacks of marginalization while retaining the advantages. It enables us to:

- Effectively undo part of the marginalization to capture the full photometric probability distribution for the pose graph bundle adjustment (section III-E).
- Update the initially visual-only marginalization prior with IMU information after the IMU initialization.
- Relinearize variables in the Markov blanket while keeping all visual and most inertial information.

The idea of delayed marginalization is that marginalization cannot be undone, but it can be delayed: *In addition* to the normal marginalization prior we also maintain a second, *delayed* marginalization prior and corresponding factor graph. In this delayed graph, marginalization of frames is performed with a delay of d. Points are still marginalized at the same time in the delayed graph, resulting in linearized photometric factors. We note that the same marginalization order as in the original graph is preserved. Switching to a fixed-lag smoother for this graph would immediately lead to a much larger Markov blanket jeopardizing the runtime of the system. E.g. in Fig. 2b we depict the delayed marginalization of $\mathbf{P}_1$. The Markov blanket only contains $\mathbf{P}_0$, $\mathbf{P}_2$, and $\mathbf{P}_3$. If we instead marginalized the oldest frame $\mathbf{P}_0$, the Markov blanket would contain $\mathbf{P}_1 - \mathbf{P}_7$, leading to higher runtime.
Marginalization in the delayed graph has the same runtime as marginalization in the original graph. The delayed graph contains the same photometric factors as the original graph, and points are marginalized at the same time. This means that each linearized photometric factor in the delayed graph is connected to exactly the $N_f = 8$ keyframes which were active when the respective factor was generated. By keeping the marginalization order, the Markov blanket in the delayed graph always has the same size as the one in the original graph. Thus, the runtime of the Schur complement is the same. This means that the overhead of Delayed Marginalization is very small even for arbitrarily large delays, as it only amounts to an additional marginalization procedure per delayed graph.

E. Pose Graph Bundle Adjustment for IMU Initialization

PGBA utilizes delayed marginalization for IMU initialization. The idea is to populate the delayed graph with IMU factors and optimize all variables (Fig. 2c).
Populating the graph: Let a frame $\mathbf{P}_i$ be directly connected to the newest pose $\mathbf{P}_k$ iff all poses $\mathbf{P}_j, i < j < k$ have not been marginalized yet. We determine the first frame $\mathbf{P}_{\text{conn}}$ in the delayed graph which is still directly connected to the newest frame. In Fig. 2c this is $\mathbf{P}_2$. From there, we insert IMU factors and bias factors to all successive frames.

We cannot start before $\mathbf{P}_{\text{conn}}$ because we do not want to insert IMU factors between non-successive keyframes. As the marginalization order is not fixed-lag, this means that we have to optimize poses without corresponding IMU variables.

It can be shown that there can be at most $N_f - 2$ poses without IMU variables. The reason is that all non-connected poses were at some point active at the same time. This means that in practice we have at least $d - N_f + 2$ poses for which we can add IMU data. In practice, we choose $N_f = 8$ and delay $d = 100$, meaning that even in the worst case there will be 93 IMU factors in the optimization. As explained previously, fixed-lag smoothing would either result in a dense Hessian or in suboptimal performance of the visual system, so this is a very good trade-off.

Optimization: We optimize the graph with the GTSAM [29] library using the Levenberg-Marquardt optimizer with the provided Ceres-default settings. In this optimization all points are marginalized. We call it pose graph bundle adjustment because it is a combination of regular pose graph optimization (PGO) and bundle adjustment (BA). In contrast to BA, we do not update our estimates for point depths and do not relinearize photometric error terms. Different from PGO we do not use binary constraints between poses, but instead use "octonary" constraints, which connect N_f frames and capture the full probability distribution of BA. Compared to PGO our solution is thus more accurate while being much faster than full BA. By using a fixed delay it is also constrained in runtime even though it can be performed at any time without losing any prior visual information.

Readvancing: Another advantage of delayed marginalization and our PGBA is that we can obtain a marginalization prior for the main system, capturing all the visual and inertial information. For this, we readvance the graph used for the PGBA. This works by successively marginalizing all the variables which have been marginalized in the main graph. Again, this is done preserving the marginalization order, which means that in each marginalization step the Markov blanket has a fixed maximum size. Hence, marginalizing step by step is significantly faster than marginalizing all variables at once, which would involve a much larger matrix inversion. Fig. 2d shows the result of readvancing.

F. Robust Multi-Stage IMU Initialization

Our initialization strategy is based on three insights:

1) When some variables are unknown (in our case scale, gravity direction, and biases) and others are close to the optimum, it is most efficient to first optimize only the unknown variables and fix the others.
2) The most accurate result can be obtained by optimizing all variables jointly, capturing the full covariance.
3) When marginalizing, connected variables have to be close to the optimum, otherwise the marginalization prior becomes inconsistent.

These observations inspire 1) the Coarse IMU Initialization, 2) the PGBA, and 3) the Marginalization Replacement (Fig. 3). Note that after "Initialize main VIO", the main VIO system III-B (green box) is already running in parallel.

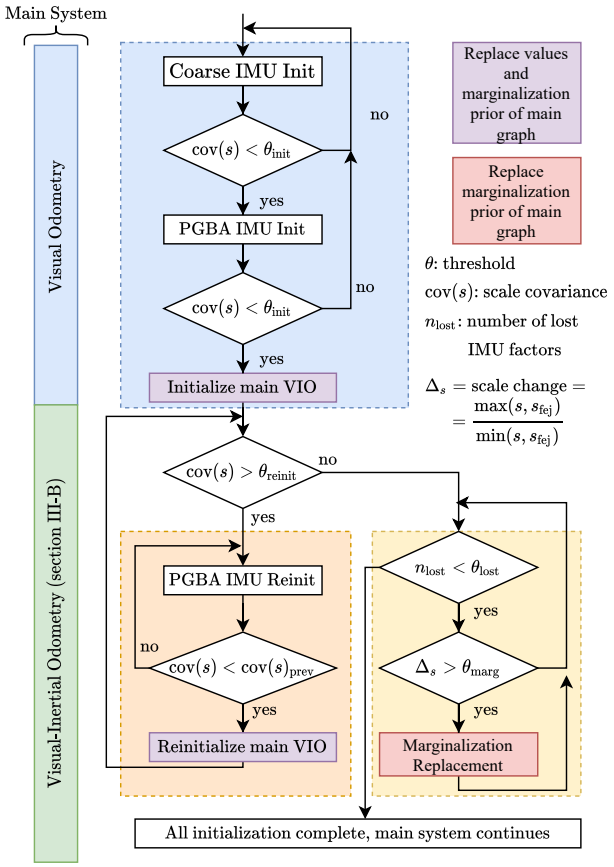


Fig. 3: Our multi-stage IMU initialization. First we perform a coarse IMU initialization, which provides initial values for the PGBA. The PGBA captures the full visual covariances, achieving very accurate initial estimates for scale, gravity direction, and biases. It also provides an updated marginalization prior for the main graph. By also optimizing the scale in the main VIO system (green box), we can initialize early (purple box) and later reinitialize or perform marginalization replacement, if new information about the scale becomes available. The proposed delayed marginalization is what enables both, the PGBA, and the marginalization replacement.

For this initializer we use a single delayed graph with a delay of $d = 100$. This delayed graph will always contain only visual factors and no IMU factors, even after the first initialization, to facilitate the marginalization replacement.

Coarse IMU Initialization: For this we only consider the last $d = 100$ keyframes and connect them with IMU factors. Similar to the inertial only optimization used for initialization in ORB-SLAM3 [5], in this optimization we fix the poses and use a single bias. We only optimize velocities, bias, the gravity direction and the scale. Gravity direction is initialized by averaging the accelerometer measurements between the first two keyframes, scale is initialized with 1, and bias and velocity with 0. This optimization is less accurate than PGBA but serves as an initialization for it. After optimizing, we compute the marginal covariance for the scale $\text{cov}(s)$ and continue to the PGBA if it is smaller

TABLE I: Evaluation of various mono (M) and stereo (S) visual-inertial odometry systems on EuRoC. Our system provides a notable improvement over the state-of-the art. Please note that a full SLAM system utilizing loop closures can achieve even more accurate results, e.g. ORB-SLAM-VI has a mean error of 0.075, and ORB-SLAM3 has a mean error of 0.043.

Sequence		MH1	MH2	MH3	MH4	MH5	V11	V12	V13	V21	V22	V23	Avg
MCSKF[2] [1] (M)	RMSE	0.42	0.45	0.23	0.37	0.48	0.34	0.20	0.67	0.10	0.16	1.13	0.414
OKVIS[1] [19] (M)	RMSE	0.33	0.37	0.25	0.27	0.39	0.094	0.14	0.21	0.090	0.17	0.23	0.231
ROVIO[2] [18] (M)	RMSE	0.21	0.25	0.25	0.49	0.52	0.10	0.10	0.14	0.12	0.14	0.14	0.224
VINS-Mono [3] (M)	RMSE	0.15	0.15	0.22	0.32	0.30	0.079	0.11	0.18	0.080	0.16	0.27	0.184
Kimera [21] (S)	RMSE	0.11	0.10	0.16	0.24	0.35	0.05	0.08	0.07	0.08	0.10	0.21	0.141
Online VIO [23] (M)	RMSE	0.14	0.13	0.20	0.22	0.20	0.05	0.07	0.16	0.04	0.11	0.17	0.135
VI-DSO [6] (M)	RMSE	**0.062**	**0.044**	0.117	0.132	0.121	0.059	0.067	0.096	0.040	0.062	0.174	0.089
	Scale Error (%)	1.1	0.5	0.4	0.2	0.8	1.1	1.1	0.8	1.2	0.3	0.4	0.7
BASALT [20] (S)	RMSE	0.07	0.06	**0.07**	0.13	0.11	**0.04**	0.05	0.10	0.04	**0.05**	-	0.072
DM-VIO (M)	RMSE	0.065	**0.044**	0.097	**0.102**	**0.096**	0.048	**0.045**	**0.069**	**0.029**	**0.050**	**0.114**	**0.069**
	Scale Error (%)	1.3	0.9	0.4	0.2	0.4	0.4	1.0	0.3	0.02	0.6	0.8	0.6

[1] results taken from [3].
[2] results taken from [30], these are $\mathbf{Sim}(3)$-aligned.
All other results are taken from the respective paper.

than a threshold θ_{init}. As shown in [31], taking into account IMU noise parameters is crucial for good IMU initialization, which our coarse IMU initialization satisfies. But for our method it is just an initialization for the PGBA, which in addition models photometric noise properties.

PGBA IMU Init.: We perform PGBA as explained in section III-E. Afterwards, we again threshold on the marginal covariance for the scale to find out if the optimization was successful. When a tighter threshold θ_{reinit} is not also met, we initialize with the result, but will perform another PGBA afterwards to reinitialize with more accurate values. This reinitialization enables us to set θ_{init} to a relatively large value, allowing to use IMU data in the main system earlier.

Marginalization Replacement: After IMU initialization, we monitor how much the scale s changes compared to the First-Estimates scale s_{fej} used in the marginalization prior. If this change exceeds a threshold θ_{marg}, i.e. $\delta_s := \max(s, s_{\text{fej}}) / \min(s, s_{\text{fej}}) > \theta_s$, we trigger a marginalization replacement. For the marginalization replacement we rebuild the PGBA graph by populating the delayed graph with IMU factors, Fig. 2c). Different from the PGBA, we do not optimize in this graph but instead just readvance it to obtain an updated marginalization prior. This new prior still contains all visual factors and at least the last $d - N_f + 1 = 93$ IMU factors. We disable the marginalization replacement if more than $\theta_{\text{lost}} = 50\%$ of the IMU factors contained in the previous prior would be lost. This procedure shows how delayed marginalization can be used to update FEJ values, overcoming one of the main problems of marginalization.

In realtime mode we perform the coarse IMU initialization and the PGBA in a separate thread. Note how important the proposed delayed marginalization is for this IMU initialization. It allows the PGBA to capture the full covariance from the photometric bundle adjustment. By readvancing, this also enables us to generate a marginalization prior for the main system, containing all IMU information from the initializer. Lastly, it is used for updating the marginalization prior when the scale changes after the initialization.

IV. Results

We evaluate our method on the EuRoC dataset [7], the TUM-VI dataset [8], and the 4Seasons dataset [9], covering flying drones, handheld sequences, and autonomous driving respectively. We encourage the reader to watch the supplementary video which shows qualitative realtime results on 4Seasons and TUM-VI slides1. We also provide ablation studies and runtime evaluations in the supplementary available at `vision.in.tum.de/dm-vio`.

Unless otherwise stated all experiments are performed in realtime mode on the same MacBook Pro 2013 (i7 at 2.3GHz) which was used for generating the results in [6], without utilizing the GPU. As ORB-SLAM3 is not officially supported on MacOS, we show results for it on a slightly stronger desktop with an Intel Core i7-7700K at 4.2GHz, which is very similar to the PC used in their paper.

All methods are evaluated 10 times for EuRoC and 5 times for the other datasets on each sequence. Following [17], results are presented in cumulative error plots, which show how many sequences (y-axis) have been tracked with an accuracy better than the threshold on the x-axis. We perform $\mathbf{SE}(3)$ alignment of the trajectory with the provided ground-truth and report the root mean squared error (RMSE), also called absolute trajectory error (ATE). On TUM-VI and 4Seasons, trajectory lengths can vary greatly so we report the drift in $\%$, which we compute with $\text{drift} = \frac{\text{rmse} \cdot 100}{\text{length}}$. We also show tables to compare to numbers from other papers and report the median result for each sequence for our method.

A. EuRoC dataset

The EuRoC dataset [7] is the most popular visual-inertial dataset to date, and many powerful methods have been eval-

TABLE II: RMSE ATE in m on the TUM-VI dataset [8]. Best results in bold, underline is the best result among monocular methods. DM-VIO outperforms even state-of-the-art stereo-inertial methods by a large margin.

Sequence	ROVIO stereo	VINS mono	OKVIS stereo	BASALT stereo	DM-VIO mono	length [m]
corridor1	0.47	0.63	0.33	0.34	**0.19**	305
corridor2	0.75	0.95	0.47	**0.42**	0.47	322
corridor3	0.85	1.56	0.57	0.35	**0.24**	300
corridor4	**0.13**	0.25	0.26	0.21	**0.13**	114
corridor5	2.09	0.77	0.39	0.37	**0.16**	270
magistrale1	4.52	2.19	3.49	**1.20**	2.35	918
magistrale2	13.43	3.11	2.73	**1.11**	2.24	561
magistrale3	14.80	**0.40**	1.22	0.74	1.69	566
magistrale4	39.73	5.12	**0.77**	1.58	1.02	688
magistrale5	3.47	0.85	1.62	**0.60**	0.73	458
magistrale6	X	2.29	3.91	3.23	**1.19**	771
outdoors1	101.95	**74.96**	X	255.04	123.24	2656
outdoors2	21.67	133.46	73.86	64.61	**12.76**	1601
outdoors3	26.10	36.99	32.38	38.26	**8.92**	1531
outdoors4	X	16.46	19.51	17.53	**15.25**	928
outdoors5	54.32	130.63	13.12	7.89	**7.16**	1168
outdoors6	149.14	133.60	96.51	65.50	**34.86**	2045
outdoors7	49.01	21.90	13.61	**4.07**	5.00	1748
outdoors8	36.03	83.36	16.31	13.53	**2.11**	986
room1	0.16	0.07	0.06	0.09	**0.03**	146
room2	0.33	**0.07**	0.11	**0.07**	0.13	142
room3	0.15	0.11	**0.07**	0.13	0.09	135
room4	0.09	0.04	**0.03**	0.05	0.04	68
room5	0.12	0.20	0.07	0.13	**0.06**	131
room6	0.05	0.08	0.04	**0.02**	**0.02**	67
slides1	13.73	0.68	0.86	0.32	**0.31**	289
slides2	0.81	0.84	2.15	**0.32**	0.87	299
slides3	4.68	0.69	2.58	0.89	**0.60**	383
avg drift%	16.83*	1.700	0.815*	0.939	**0.472**	normalized

uated on it. In Table I we compare to the state-of-the art in visual-inertial odometry, all results are without loop-closure. Our method outperforms all other methods clearly in terms of RMSE. The closest competitor is Basalt [20], a stereo-inertial method which achieves a smaller error on 2 sequences. We also observe the lowest average scale error reported on the dataset so far, confirming that our contributions in IMU initialization have a positive impact on performance. In the supplementary we provide runtime evaluations, showing that tracking takes 10.34ms on average, and keyframe processing takes 53.67ms. The delayed marginalization is responsible for an overhead of 0.44ms or 0.8% in the keyframe thread.

B. TUM-VI dataset

The TUM-VI dataset [8] is a very challenging handheld dataset, featuring large-scale indoor and outdoor scenes, and even sequences sliding down a tube, where almost the full image is covered. With long periods of walking in straight lines, stereo methods have an advantage here as they still can observe the scale with constant motion. We compare to the state-of-the-art visual-inertial odometry methods evaluated in [8] in Table II. Our method clearly outperforms the other monocular method in VINS-Mono [3] on most sequences, and even compared to the stereo methods it shows the best result on 16 sequences and a mean drift of 0.472. The closest competitor is again Basalt, which achieves the best result on 8 sequences and a mean drift of 0.939.

On this dataset, we also evaluate against ORB-SLAM3 [5], which is the state-of-the art visual-inertial SLAM system. This is not entirely fair as ORB-SLAM3 uses loop closures

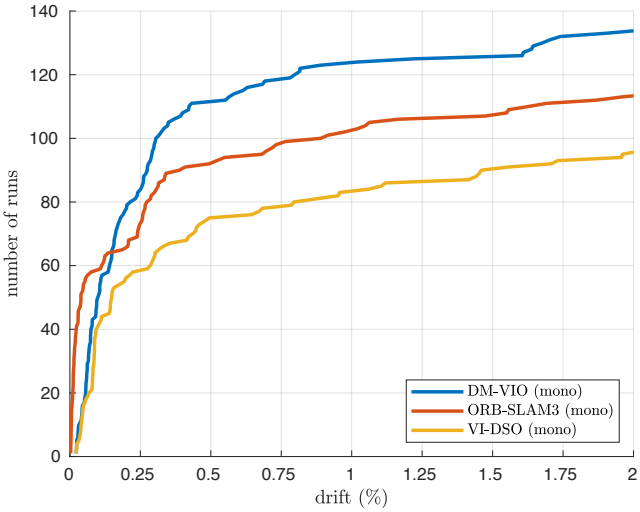


Fig. 4: Cumulative error plot for the TUM-VI dataset (drift in %). Our method clearly outperforms both VI-DSO and ORB-SLAM3 in terms of robustness. Thanks to its powerful loop closure system, ORB-SLAM3 has an advantage in terms of accuracy on some sequences.

(which cannot be disabled), constituting an advantage over the other methods. We find the comparison still helpful as it allows to make conclusions regarding the underlying odometry. We have evaluated ORB-SLAM3 5 times on each sequence and reproduced their results with code and settings provided by the authors. For this comparison we have also evaluated VI-DSO [6], and the results are shown in Fig. 4. We observe that ORB-SLAM3 is more accurate on some sequences thanks to its very strong loop closure system. However, our method is more robust overall. This indicates that an integration of loop closure and map reuse into our system would be an interesting future research direction.

C. 4Seasons dataset

The 4Seasons dataset [9] is a very recent automotive dataset, which, in contrast to most other car datasets, features a well time-synchronized visual-inertial sensor. The lower part of the images is obstructed by the car hood, hence we crop off the bottom 96 pixels, which we do for all methods. As this is the first odometry method to evaluate on the 4Seasons dataset, we make sure to determine IMU noise parameters for all methods the same way to ensure a fair comparison: We have manually read off the accelerometer and gyroscope noise density and bias random walk from the Allan variance plot provided in the data sheet of the IMU. To handle unmodeled effects we follow [8] and inflate noise values by different amounts to determine the best setting for all methods. For each method we tried noise models inflated by 1, 10, 100, 1000 respectively and chose the configuration which gave best results. For VI-DSO and for our method we slightly modified the visual initializer by adding a zero-prior to the translation on the x and y axis, and also added a threshold to stop keyframe creation for translations

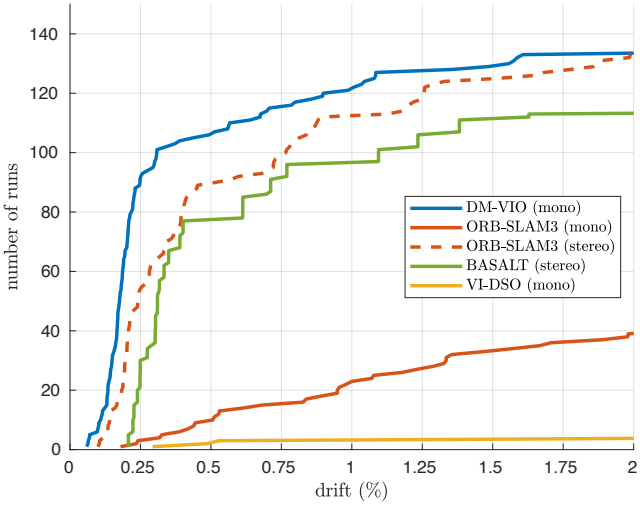


Fig. 5: Cumulative error plot for the 4Seasons dataset (drift in %). With lots of stretches with constant velocity, this dataset is extremely challenging for monocular visual-inertial methods. Thanks to our novel IMU initializer powered by delayed marginalization and PGBA, DM-VIO is able to cope with it and even outperforms stereo-inertial methods.

smaller than 0.01m (the latter was not activated for VI-DSO as it did not improve the results for it). Otherwise, parameters are the same as for the other experiments. For Basalt we tried all three provided default configurations with the optimal noise values to find the best settings. After choosing the configuration for each method, we perform one final evaluation, running all 30 sequences 5 times each.

The results are shown in Fig. 5. It is clear that the automotive scenario is very challenging for monocular methods. This is expected as it naturally features many stretches with constant motion, where scale is not observable, constituting a challenge for IMU initialization. Thanks to our novel IMU initialization, DM-VIO not only works well on the dataset but even outperforms stereo-inertial ORB-SLAM3 and Basalt, while using monocular images and no loop closures.

V. Conclusion and Future Work

We have presented a monocular visual-inertial odometry system which outperforms the state of the art, even stereo-inertial methods. Thanks to a novel IMU initializer, it works well in flying, handheld, and automotive scenarios, extending the applicability of monocular methods. The foundation of our IMU initialization is delayed marginalization, which also enables the pose graph bundle adjustment.

We anticipate that this method will spark further research in this direction. The idea of delayed marginalization could be applied to more use cases, e.g. for reactivating old keyframes in a marginalization setting to enable map reuse. The pose graph bundle adjustment can also be applied to long-term loop closures. Lastly, our open-source system is easily extendible, as all optimizations are integrated with GTSAM, allowing to quickly add new factors. This could be used for GPS integration, wheel odometry, and more.

References

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *ICRA*, 2007.

[2] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *RA-L*, vol. 2, no. 1, 2017.

[3] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *T-RO*, vol. 34, no. 4, 2018.

[4] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *RA-L*, vol. 2, no. 2, 2017.

[5] C. Campos, R. Elvira, J. J. G. Rodrguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap slam," *T-RO*, pp. 1–17, 2021.

[6] L. von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *ICRA*, May 2018.

[7] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *IJRR*, 2016.

[8] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *IROS*, October 2018.

[9] P. Wenzel, R. Wang, N. Yang, Q. Cheng, Q. Khan, L. von Stumberg, N. Zeller, and D. Cremers, "4Seasons: A cross-season dataset for multi-weather SLAM in autonomous driving," in *GCPR*, 2020.

[10] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *CVPR*, vol. 1, June 2004, pp. 652–659.

[11] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *TPAMI*, vol. 29, 2007.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular slam system," *T-RO*, vol. 31, Oct 2015.

[14] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *ICRA*, 2013.

[15] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *ICCV*, 2011.

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[17] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *TPAMI*, vol. 40, 2018.

[18] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IROS*, 2015.

[19] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *IJRR*, 2014.

[20] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *RA-L*, vol. 5, no. 2, pp. 422–429, 2020.

[21] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *ICRA*, 2020.

[22] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *IJCV*, vol. 106, no. 2, 2014.

[23] E. Hong and J. Lim, "Visual-inertial odometry with robust initialization and online scale estimation," *Sensors*, vol. 18, p. 4287, 12 2018.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *RSS*, 2015.

[25] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers," in *CRV*, 2015.

[26] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *T-RO*, vol. 28, no. 1, pp. 61–76, 2012.

[27] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *ICRA*, 2014.

[28] G. Huang, A. I. Mourikis, and S. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency," in *ISER*, 2008.

[29] F. Daellert and Others, "Gtsam," https://gtsam.org.

[30] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *ICRA*, 2018, pp. 2502–2509.

[31] C. Campos, J. Montiel, and J. D. Tardós, "Inertial-only optimization for visual-inertial initialization," *ICRA*, pp. 51–57, 2020.

D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry

Nan Yang[1,2] Lukas von Stumberg[1,2] Rui Wang[1,2] Daniel Cremers[1,2]
[1] Technical University of Munich [2] Artisense

Abstract

We propose D3VO as a novel framework for monocular visual odometry that exploits deep networks on three levels – deep depth, pose and uncertainty estimation. We first propose a novel self-supervised monocular depth estimation network trained on stereo videos without any external supervision. In particular, it aligns the training image pairs into similar lighting condition with predictive brightness transformation parameters. Besides, we model the photometric uncertainties of pixels on the input images, which improves the depth estimation accuracy and provides a learned weighting function for the photometric residuals in direct (feature-less) visual odometry. Evaluation results show that the proposed network outperforms state-of-the-art self-supervised depth estimation networks. D3VO tightly incorporates the predicted depth, pose and uncertainty into a direct visual odometry method to boost both the front-end tracking as well as the back-end non-linear optimization. We evaluate D3VO in terms of monocular visual odometry on both the KITTI odometry benchmark and the EuRoC MAV dataset. The results show that D3VO outperforms state-of-the-art traditional monocular VO methods by a large margin. It also achieves comparable results to state-of-the-art stereo/LiDAR odometry on KITTI and to the state-of-the-art visual-inertial odometry on EuRoC MAV, while using only a single camera.

1. Introduction

Deep learning has swept most areas of computer vision – not only high-level tasks like object classification, detection and segmentation [30, 39, 58], but also low-level ones such as optical flow estimation [12, 65] and interest point detection and description [11, 13, 79]. Yet, in the field of Simultaneously Localization And Mapping (SLAM) or Visual Odometry (VO) which estimates the relative camera poses from image sequences, traditional geometric-based approaches [16, 17, 53] still dominate the field. While monocular methods [16,52] have the advantage of low hardware cost and less calibration effort, they cannot achieve competitive performance compared to stereo [53, 74] or visual-inertial odometry (VIO) [44, 54, 56, 72], due to the scale drift [62, 77] and low robustness. Recently, there have been many efforts to address this by leveraging deep neural networks [48, 68, 80, 83]. It has been shown that with deep monocular depth estimation networks [26, 27, 43, 78], the performance of monocular VO is boosted, since deep networks are able to estimate depth maps with consistent metric scale by learning a-priori knowledge from a large amount of data [42].

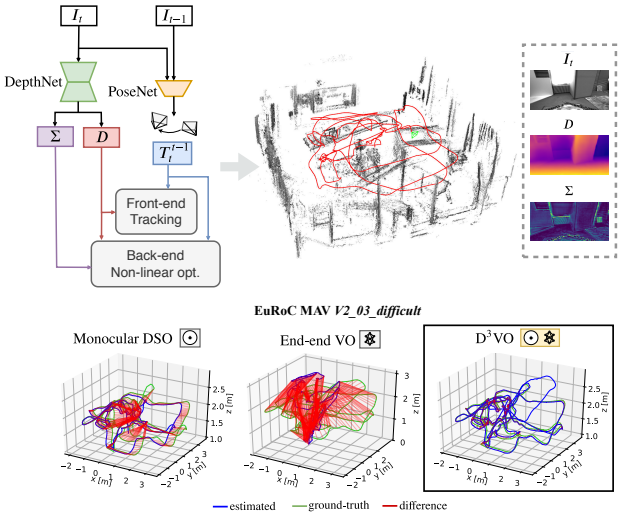


Figure 1: We propose D3VO – a novel monocular visual odometry (VO) framework which exploits deep neural networks on three levels: **D**eep depth (D), **D**eep pose (T_t^{t-1}) and **D**eep uncertainty (Σ) estimation. D3VO integrates the three estimations tightly into both the front-end tracking and the back-end non-linear optimization of a sparse direct odometry framework [16].

In this way, however, deep neural networks are only used to a limited degree. Recent advances of self- and unsupervised monocular depth estimation networks [26, 86] show that the poses of the adjacent monocular frames can be predicted together with the depth. Since the pose estimation from deep neural networks shows high robustness, one question arises: *Can the deep-predicted poses be employed to boost traditional VO*? On the other hand, since

SLAM/VO is essentially a state estimation problem where uncertainty plays an important role [19, 63, 69] and meanwhile many learning based methods have started estimating uncertainties, the next question is, *how can we incorporate such uncertainty-predictions into optimization-based VO*?

In this paper, we propose D3VO as a framework for monocular direct (feature-less) visual VO that exploits self-supervised monocular depth estimation network on three levels: *deep depth*, *pose* and *uncertainty* estimation, as shown in Fig. 1. To this end, we first propose a purely self-supervised network trained with stereo videos. The proposed self-supervised network predicts the depth from a single image with DepthNet and the pose between two adjacent frames with PoseNet. The two networks are bridged by minimizing the photometric error originated from both *static* stereo warping with the rectified baseline and *temporal* warping using the predicted pose. In this way, the temporal information is incorporated into the training of depth, which leads to more accurate estimation. To deal with the inconsistent illumination between the training image pairs, our network predicts the *brightness transformation parameters* which align the brightness of source and target images during training on the fly. The evaluation on the EuRoC MAV dataset shows that the proposed brightness transformation significantly improves the depth estimation accuracy. To integrate the deep depth into VO system, we firstly initialize every new 3D point with the predicted depth with a metric scale. Then we adopt the *virtual stereo term* proposed in Deep Virtual Stereo Odometry (DVSO) [78] to incorporate the predicted pose into the non-linear optimization. Unlike DVSO which uses a semi-supervised monocular depth estimation network relying on auxiliary depth extracted from state-of-the-art stereo VO system [74], our network uses only stereo videos without any external depth supervision.

Although the illumination change is explicitly modeled, it is not the only factor which may violate the brightness constancy assumption [40]. Other factors, e.g., non-Lambertian surfaces, high-frequency areas and moving objects, also corrupt it. Inspired by the recent research on aleatoric uncertainty by deep neural networks [35, 40], the proposed network estimates the photometric uncertainty as predictive variance conditioned on the input image. As a result, the errors originated from pixels which are likely to violate the brightness constancy assumption are down-weighted. The learned weights of the photometric residuals also drive us to the idea of incorporating it into direct VO – since both the self-supervised training scheme and the direct VO share a similar photometric objective, we propose to use the learned weights to replace the weighting function of the photometric residual in traditional direct VO which is empirically set [61] or only accounts for the intrinsic uncertainty of the specific algorithm itself [16, 37].

Robustness is one of the most important factors in designing VO algorithm. However, traditional monocular visual VO suffers from a lack of robustness when confronted with low textured areas or fast movement [72]. The typical solution is to introduce an inertial measurement unit (IMU). But this increases the calibration effort and, more importantly, at constant velocity, IMUs cannot deliver the metric scale in constant velocity [50]. We propose to increase the robustness of monocular VO by incorporating the estimated pose from the deep network into both the front-end tracking and the back-end non-linear optimization. For the front-end tracking, we replace the pose from the constant velocity motion model with the estimated pose from the network. Besides, the estimated pose is also used as a squared regularizer in addition to direct image alignment [66]. For the back-end non-linear optimization, we propose a pose energy term which is jointly minimized with the photometric energy term of direct VO.

We evaluate the proposed monocular depth estimation network and D3VO on both KITTI [25] and EuRoC MAV [5]. We achieve state-of-the-art performances on both monocular depth estimation and camera tracking. In particular, by incorporating deep depth, deep uncertainty and deep pose, D3VO achieves comparable results to state-of-the-art stereo/LiDAR methods on KITTI Odometry, and also comparable results to the state-of-the-art VIO methods on EuRoC MAV, while being a monocular method.

2. Related Work

Deep learning for monocular depth estimation. Supervised learning [15, 43, 45] shows great performance on monocular depth estimation. Eigen et al. [14, 15] propose to use multi-scale CNNs which directly regresses the pixel-wise depth map from a single input image. Laina et al. [43] propose a robust loss function to improve the estimation accuracy. Fu et al. [24] recast the monocular depth estimation network as an ordinal regression problem and achieve superior performance. More recent works start to tackle the problem in a self- and unsupervised way by learning the depth map using the photometric error [27, 28, 49, 73, 81, 82, 86] and adopting differentiable interpolation [32]. Our self-supervised depth estimation network builds upon MonoDepth2 [26] and extends it by predicting the brightness transformation parameters and the photometric uncertainty.

Deep learning for uncertainty estimation. The uncertainty estimation of deep learning has recently been investigated in [35, 36] where two types of uncertainties are proposed. Klodt et al. [40] propose to leverage the concept of aleatoric uncertainty to estimate the photometric and the depth uncertainties in order to improve the depth estimation accuracy. However, when formulating the photometric uncertainty, they do not consider brightness changes

across different images which in fact can be modeled explicitly. Our method predicts the photometric uncertainty conditioned on the brightness-aligned image, which can deliver better photometric uncertainty estimation. Besides, we also seek to make better use of our learned uncertainties and propose to incorporate them into traditional VO systems [16].

Deep learning for VO / SLAM. End-to-end learned deep neural networks have been explored to directly predict the relative poses between images with supervised [70, 75, 85] or unsupervised learning [46, 73, 82, 86]. Besides pose estimation, CodeSLAM [2] delivers dense reconstruction by jointly optimizing the learned prior of the dense geometry together with camera poses. However, in terms of pose estimation accuracy all these end-to-end methods are inferior to classical stereo or visual inertial based VO methods. Building on the success of deep monocular depth estimation, several works integrate the predicted depth/disparity map into monocular VO systems [68, 78] to improve performance and eliminate the scale drift. CNN-SLAM [68] fuses the depth predicted by a supervised deep neural network into LSD-SLAM [17] and the depth maps are refined with Bayesian filtering, achieving superior performance in indoor environments [29, 64]. Other works [10, 67] explore the application of deep neural networks on feature based methods ,and [34] uses Generative Adversarial Networks (GANs) as an image enhancement method to improve the robustness of VO in low light. The most related work to ours is Deep Virtual Stereo Odometry (DVSO). DVSO proposes a virtual stereo term that incooperates the depth estimation from a semi-supervised network into a direct VO pipeline. In particular, DVSO outperforms other monocular VO systems by a large margin, and even achieves comparable performance to state-of-the-art stereo visual odometry systems [53, 74]. While DVSO merely leverages the depth, the proposed D3VO exploits the power of deep networks on multiple levels thereby incorporating more information into the direct VO pipeline.

3. Method

We first introduce a novel self-supervised neural network that predicts depth, pose and uncertainty. The network also estimates *affine brightness transformation parameters* to align the illumination of the training images in a self-supervised manner. The photometric uncertainty is predicted based on a distribution over the possible brightness values [35, 40] for each pixel. Thereafter we introduce D3VO as a direct visual odometry framework that incorporates the predicted properties into both the tracking front-end and the photometric bundle adjustment backend.

3.1. Self-supervised Network

The core concept of the proposed monocular depth estimation network is the self-supervised training scheme

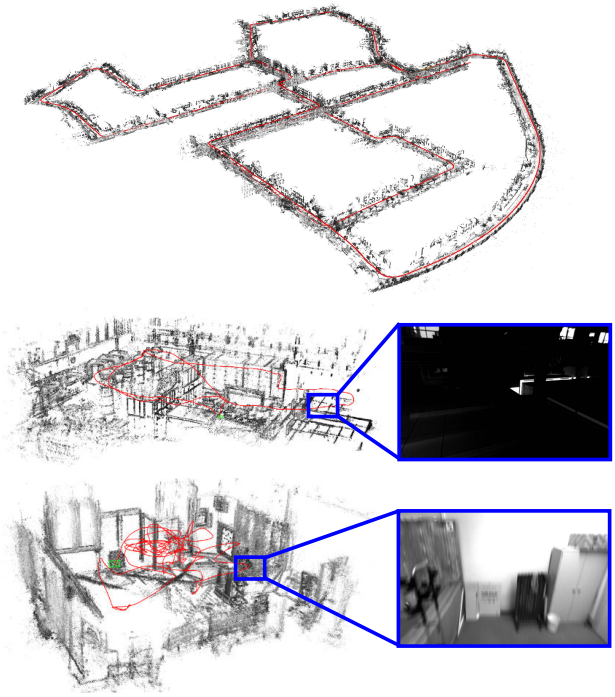


Figure 2: Examples of point clouds and trajectories delivered by D3VO on KITTI Odometry Seq. 00, EuRoC *MH_05_difficult* and *V1_03_difficult*. The insets on EuRoC show the scenarios with low illumination and motion blur which are among the main reasons causing failures of traditional purely vision-based VO systems.

which simultaneously learns depth with DepthNet and motion with PoseNet using video sequences [26, 86]. The self-supervised training is realized by minimizing the minimum of the photometric re-projection errors between the temporal and static stereo images:

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t, I_{t' \to t}). \tag{1}$$

where V is the set of all pixels on I_t and t' is the index of all source frames. In our setting I_t is the left image and $I_{t'}$ contains its two adjacent temporal frames and its opposite (right) frame, i.e., $I_{t'} \in \{I_{t-1}, I_{t+1}, I_{t^s}\}$. The per-pixel minimum loss is proposed in Monodepth2 [26] in order to handle the occlusion among different source frames. To simplify notation, we use I instead of $I(\mathbf{p})$ in the remainder of this section. $I_{t' \to t}$ is the sythesized I_t by warping the temporal stereo images with the predicted depth D_t, the camera pose $\mathbf{T}_t^{t'}$, the camera intrinsics K, and the differentialble bilinear sampler [32]. Note that for $I_{t^s \to t}$, the transformation $\mathbf{T}_t^{t^s}$ is known and constant. DepthNet also predicts the depth map D_{t^s} of the right image I_{t^s} by feeding only the left image I_t as proposed in [27]. The training of D_{t^s} requires to synthesize $I_{t \to t^s}$ and compare with I_{t^s}. For simplicity, we will in the following only detail the loss

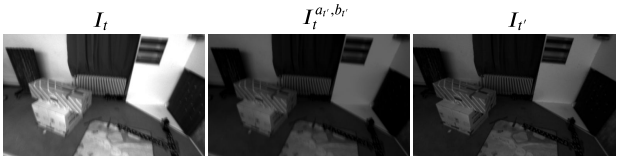


Figure 3: Examples of affine brightness transformation on EuRoC MAV [5]. Originally the source image ($I_{t'}$) and the target image (I_t) show different brightness. With the predicted parameters a, b, the transformed target images $I^{a',b'}$ have similar brightness as the source images, which facilitates the self-supervised training based on the brightness constancy assumption.

regarding the left image.

The common practice [27] is to formulate the photometric error as

$$r(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1-\alpha)||I_a - I_b||_1 \quad (2)$$

based on the brightness constancy assumption. However, it can be violated due to illumination changes and auto-exposure of the camera to which both L1 and SSIM [76] are not invariant. Therefore, we propose to explicitly model the camera exposure change with predictive *brightness transformation parameters*.

Brightness transformation parameters. The change of the image intensity due to the adjustment of camera exposure can be modeled as an affine transformation with two parameters a, b

$$I^{a,b} = aI + b. \quad (3)$$

Despite its simplicity, this formulation has been shown to be effective in direct VO/SLAM, e.g., [16, 18, 33, 74], which builds upon the brightness constancy assumption as well. Inspired by these works, we propose predicting the transformation parameters a, b which align the brightness condition of I_t with $I_{t'}$. We reformulate Eq. (1) as

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t^{a_{t'},b_{t'}}, I_{t' \to t}) \quad (4)$$

with

$$I_t^{a_{t'},b_{t'}} = a_{t \to t'} I_t + b_{t \to t'}, \quad (5)$$

where $a_{t \to t'}$ and $b_{t \to t'}$ are the transformation parameters aligning the illumination of I_t to $I_{t'}$. Note that both parameters can be trained in a self-supervised way without any supervisional signal. Fig. 3 shows the affine transformation examples from EuRoC MAV [5].

Photometric uncertainty. Only modeling affine brightness change is not enough to capture all failure cases of the brightness constancy assumption. Other cases like non-Lambertian surfaces and moving objects, are caused by the intrinsic properties of the corresponding objects which are not trivial to model analytically [40]. Since these aspects can be seen as observation noise, we leverage the concept of heteroscedastic aleatoric uncertainty of deep neural networks proposed by Kendall et al. [35]. The key idea is to predict a posterior probability distribution for each pixel parameterized with its mean as well as its variance $p(y|\tilde{y}, \sigma)$ over ground-truth labels y. For instance, by assuming the noise is Laplacian, the negative log-likelihood to be minimized is

$$-\log p(y|\tilde{y}, \sigma) = \frac{|y - \tilde{y}|}{\sigma} + \log \sigma + const. \quad (6)$$

Note that no ground-truth label for σ is needed for training. The predictive uncertainty allows the network to adapt the weighting of the residual dependent on the data input, which improves the robustness of the model to noisy data or erroneous labels [35].

In our case where the "ground-truth" y are the pixel intensities on the target images, the network will predict higher σ for the pixel areas on I_t where the brightness constancy assumption may be violated. Similar to [40], we implement this by converting Eq. (4) to

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \frac{\min_{t'} r(I_t^{a_{t'},b_{t'}}, I_{t' \to t})}{\Sigma_t} + \log \Sigma_t, \quad (7)$$

where Σ_t is the uncertainty map of I_t. Fig. 4 shows the qualitative results of the predicted uncertainty maps on KITTI [25] and EuRoC [5] datasets, respectively. In the next section, we will show that the learned Σ_t is useful for weighting the photometric residuals for D3VO.

The total loss function is the summation of the self-supervised losses and the regularization losses on multi-scale images:

$$L_{total} = \frac{1}{s} \sum_{s} (L_{self}^s + \lambda L_{reg}^s), \quad (8)$$

where $s = 4$ is the number of scales and

$$L_{reg} = L_{smooth} + \beta L_{ab} \quad (9)$$

with

$$L_{ab} = \sum_{t'} (a_{t'} - 1)^2 + b_{t'}^2 \quad (10)$$

is the regularizer of the brightness parameters and L_{smooth} is the edge-aware smoothness on D_t [27].

To summarize, the proposed DepthNet predicts D_t, D_{t^s} and Σ_t with one single input I_t. PoseNet predicts $\mathbf{T}_t^{t'}$, $a_{t \to t'}$ and $b_{t \to t'}$ with channel-wise concatenated (I_t, $I_{t'}$) as the input. Both DepthNet and PoseNet are convolutional networks following the widely used UNet-like architecture [59]. Please refer to our supplementary materials for network architecture and implementation details.

3.2. D3VO

In the previous section, we introduced the self-supervised depth estimation network which predicts the depth map D, the uncertainty map Σ and the relative pose $\mathbf{T}_t^{t'}$. In this section, we will describe how D3VO integrates these predictions into a windowed sparse photometric bundle adjustment formulation as proposed in [16]. Note that

in the following we use $\widetilde{\cdot}$ denoting the predictions from the network as $\widetilde{D}$, $\widetilde{\Sigma}$ and $\widetilde{\mathbf{T}}_t^{t'}$ to avoid ambiguity.

Photometric energy. D3VO aims to minimize a total photometric error E_{photo} defined as

$$E_{photo} = \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in \text{obs}(\mathbf{p})} E_{\mathbf{p}j}, \tag{11}$$

where $\mathcal{F}$ is the set of all keyframes, $\mathcal{P}_i$ is the set of points hosted in keyframe i, obs($\mathbf{p}$) is the set of keyframes in which point $\mathbf{p}$ is observable and $E_{\mathbf{p}j}$ is the weighted photometric energy term when $\mathbf{p}$ is projected onto keyframe j:

$$E_{\mathbf{p}j} := \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} w_\mathbf{p} \left|\left| (I_j[\mathbf{p}'] - b_j) - \frac{e^{a_j}}{e^{a_i}}(I_i[\mathbf{p}] - b_i) \right|\right|_\gamma, \tag{12}$$

where $\mathcal{N}$ is the set of 8 neighboring pixels of $\mathbf{p}$ defined in [16], a,b are the affine brightness parameters jointly estimated by non-linear optimization as in [16] and $||\cdot||_\gamma$ is the Huber norm. In [16], the residual is down-weighted when the pixels are with high image gradient to compensate small independent geometric noise [16]. In realistic scenarios, there are more sources of noise, e.g., reflection [40], that need to be modeled in order to deliver accurate and robust motion estimation. We propose to use the learned uncertainty $\widetilde{\Sigma}$ to formulate the weighting function

$$w_\mathbf{p} = \frac{\alpha^2}{\alpha^2 + \left|\left|\widetilde{\Sigma}(\mathbf{p})\right|\right|_2^2}, \tag{13}$$

which may not only depend on local image gradient, but also on higher level noise pattern. As shown in Fig. 4, the proposed network is able to predict high uncertainty on the areas of reflectance, e.g., the windows of the vehicles, the moving object like the cyclist and the object boundaries where depth discontinuity occurs.

The projected point position of $\mathbf{p}'$ is given by $\mathbf{p}' = \Pi(\mathbf{T}_i^j \Pi^{-1}(\mathbf{p}, d_\mathbf{p}))$, where $d_\mathbf{p}$ is the depth of the point $\mathbf{p}$ in the coordinate system of keyframe i and $\Pi(\cdot)$ is the projection function with the known camera intrinsics. Instead of randomly initializing $d_\mathbf{p}$ as in traditional monocular direct methods [16, 17], we initialize the point with $d_\mathbf{p} = \widetilde{D}_i[\mathbf{p}]$ which provides the metric scale. Inspired by [78], we introduce a *virtual stereo term* $E_\mathbf{p}^\dagger$ to Eq. (11)

$$E_{photo} = \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \left(\lambda E_\mathbf{p}^\dagger + \sum_{j\in \text{obs}(\mathbf{p})} E_{\mathbf{p}j} \right) \tag{14}$$

with

$$E_\mathbf{p}^\dagger = w_\mathbf{p} \left|\left| I_i^\dagger[\mathbf{p}^\dagger] - I_i[\mathbf{p}] \right|\right|_\gamma, \tag{15}$$

$$I_i^\dagger[\mathbf{p}^\dagger] = I_i[\Pi(\mathbf{T_s}^{-1} \Pi^{-1}(\mathbf{p}^\dagger, D_{i^s}[\mathbf{p}^\dagger]))] \tag{16}$$

with $\mathbf{T_s}$ the transformation matrix from the left to the right image used for training DepthNet and

$$\mathbf{p}^\dagger = \Pi(\mathbf{T_s} \Pi^{-1}(\mathbf{p}, d_\mathbf{p})). \tag{17}$$

The virtual stereo term optimizes the estimated depth $d_\mathbf{p}$ from VO to be consistent with the depth predicted by the proposed deep network [78].

Pose energy. Unlike traditional direct VO approaches [19, 23] which initialize the front-end tracking for each new frame with a constant velocity motion model, we leverage the predicted poses between consecutive frames to build a non-linear factor graph [41, 47]. Specifically, we create a new factor graph whenever the newest keyframe, which is also the reference frame for the front-end tracking, is updated. Every new frame is tracked with respect to the reference keyframe with direct image alignment [66]. Additionally, the predicted relative pose from the deep network is used as a factor between the current frame and the last frame. After the optimization is finished, we marginalize the last frame and the factor graph will be used for the front-end tracking of the following frame. Please refer to our supp. materials for the visualization of the factor graph.

The pose estimated from the tracking front-end is then used to initialize the photometric bundle adjustment backend. We further introduce a prior for the relative keyframe pose $\mathbf{T}_{i-1}^i$ using the predicted pose $\widetilde{\mathbf{T}}_{i-1}^i$. Note that $\widetilde{\mathbf{T}}_{i-1}^i$ is calculated by concatenating all the predicted frame-to-frame poses between keyframe $i-1$ and i. Let

$$E_{pose} = \sum_{i\in\mathcal{F}-\{0\}} \text{Log}(\widetilde{\mathbf{T}}_{i-1}^i \mathbf{T}_i^{i-1})^\top \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\xi}}_{i-1}^i}^{-1} \text{Log}(\widetilde{\mathbf{T}}_{i-1}^i \mathbf{T}_i^{i-1}), \tag{18}$$

where Log: SE(3) $\rightarrow \mathbb{R}^6$ maps from the transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$ in the Lie group SE(3) to its corresponding twist coordinate $\boldsymbol{\xi} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$. The diagonal inverse covariance matrix $\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\xi}}_{i-1}^i}^{-1}$ is obtained by propagating the covariance matrix between each consecutive frame pairs that is modeled as a constant diagonal matrix.

The total energy function is defined as

$$E_{total} = E_{photo} + wE_{pose}. \tag{19}$$

Including the pose prior term E_{pose} in Eq. 19 can be considered as an analogy to integrating the pre-integrated IMU pose prior into the system with a Gaussian noise model. E_{total} is minimized using the Gauss-Newton method. To summarize, we boost the direct VO method by introducing the predicted poses as initializations to both the tracking front-end and the optimization backend, as well as adding them as a regularizer to the energy function of the photometric bundle adjustment.

4. Experiments

We evaluate the proposed self-supervised monocular depth estimation network as well as D3VO on both the KITTI [25] and the EuRoC MAV [5] datasets.

Approach	Train	RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
MonoDepth2 [27]	MS	4.750	0.196	0.106	0.818	0.874	*0.957*	0.979
Ours, *uncer*	MS	*4.532*	*0.190*	*0.101*	*0.772*	*0.884*	0.956	*0.978*
Ours, *ab*	MS	4.650	0.193	0.105	0.791	0.878	*0.957*	0.979
Ours, *full*	MS	**4.485**	**0.185**	**0.099**	**0.763**	**0.885**	**0.958**	**0.979**
Kuznietsov et al. [42]	DS	4.621	0.189	0.113	*0.741*	0.862	**0.960**	**0.986**
DVSO [78]	D*S	**4.442**	*0.187*	**0.097**	**0.734**	**0.888**	0.958	*0.980*
Ours	MS	*4.485*	**0.185**	*0.099*	0.763	*0.885*	0.958	0.979

Table 1: Depth evaluation results on the KITTI Eigen split [15]. M: self-supervised monocular supervision; S: self-supervised stereo supervision; D: ground-truth depth supervison; D*: sparse auxiliary depth supervision. The upper part shows the comparison with the SOTA self-supervised network Monodepth2 [26] under the same setting and the ablation study of the brightness transformation parameters (*ab*) and the photometric uncertainty (*uncer*). The lower part shows the comparison with the SOTA *semi*-supervised methods using stereo as well as depth supervision. Our method outperforms Monodepth2 on all metrics and can also deliver comparable performance to the SOTA semi-supervised method DVSO [78] that additionally uses the depth from Stereo DSO [74] as sparse supervision signal.

4.1. Monocular Depth Estimation

KITTI. We train and evalutate the proposed self-supervised depth estimation network on the split of Eigen at el. [15]. The network is trained on stereo sequences with the pre-processing proposed by Zhou et al. [86], which gives us 39,810 training quadruplets, each of which contains 3 (left) temporal images and 1 (right) stereo image, and 4,424 for validation. The upper part of Table 1 shows the comparison with Monodepth2 [26] which is the state-of-the-art method trained with stereo and monocular setting, and also the ablation study of the proposed brightness transformation prediction (*ab*) and the photometric uncertainty estimation (*uncer*). The results demonstrate that the proposed depth estimation network outperforms Monodepth2 on all metrics. The ablation studies unveil that the significant improvement over Monodepth2 comes largely with *uncer*, possibly because in KITTI there are many objects with non-Lambertian surfaces like windows and also objects that move independently such as cars and leaves which violate the brightness constancy assumption. The lower part of the table shows the comparison to the state-of-the-art *semi*-supervised methods and the results show that our method can achieve competitive performance without using any depth supervision.

In Figure 4 we show some qualitative results obtained from the Eigen test set [15]. From left to right, the original image, the depth maps and the uncertainty maps are shown respectively. For more qualitative results and the generalization capability on the Cityscapes dataset [8], please refer to our supp. materials.

EuRoC MAV. The EuRoC MAV Dataset [5] is a dataset containing 11 sequences categorized as *easy*, *medium* and *difficult* according to the illumination and camera motion. This dataset is very challenging due to the strong motion and significant illumination changes both between stereo and temporal images. We therefore consider it as a nice test bench for validating the effectiveness of our predictive brightness transformation parameters for depth prediction. Inspired by Gordon et al. [28] who recently generated ground truth depth maps for the sequence *V2_01* by projecting the provided Vicon 3D scans and filtering out occluded points, we also use this sequence for depth evaluations[1]. Our first experiment is set up to be consistent as in [28], for which we train models with the monocular setting on all

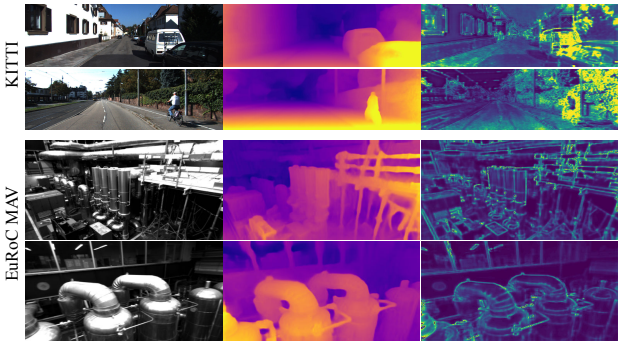


Figure 4: Qualitative results from KITTI and EuRoC MAV. The original image, the predicted depth maps and the uncertainty maps are shown from the left to the right, respectively. In particular, the network is able to predict high uncertainty on object boundaries, moving objects, highly reflecting and high frequency areas.

	RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$
Monodepth2	0.370	0.148	0.102	0.065	0.890
Ours, *ab*	*0.339*	*0.130*	*0.086*	*0.054*	*0.929*
Ours, *uncer*	0.368	0.144	0.100	0.065	0.892
Ours, *full*	**0.337**	**0.128**	**0.082**	**0.051**	**0.931**

Table 2: Evaluation results of *V2_01* in EuRoC MAV [5]. The performance of monocular depth estimation is boosted largely by the proposed predictive brightness transformation parameters.

	RMSE	RMSE (log)	ARD	SRD	$\delta < 1.25$
[28]	0.971	0.396	0.332	0.389	0.420
Ours	**0.943**	**0.391**	**0.330**	**0.375**	**0.438**

Table 3: Evaluation results of *V2_01* in EuRoC MAV [5] with the model trained with all *MH* sequences.

[1] We thank the authors of [28] to provide the processing code.

MH sequences and test on *V2_01* and show the results in Table 3.

In the second experiment, we use 5 sequences *MH_01*, *MH_02*, *MH_04*, *V1_01* and *V1_02* as the training set to check the performance of our method in a relatively loosened setting. We remove the static frames for training and this results in 12,691 images of which 11,422 images are used for training and 1269 images are used for validation. We train our model with different ablations, as well as Monodepth2 [26] as the baseline. The results in Table 2 show that all our variations outperform the baseline and, in contrast to the case in KITTI, the proposed *ab* improves the results on this dataset significantly. Please refer to the supp. materials for more experiments on *ab*. In fact, it is worth noting that the results in Table 3 (trained on one scene *MH* and tested on another scene *V*) are worse than the ones in Table 2 (trained on both *MH* and *V*), which implies that it is still a challenge to improve the generalization capability of monocular depth estimation among very different scenarios.

4.2. Monocular Visual Odometry

We evaluate the VO performance of D3VO on both KITTI Odometry and EuRoC MAV with the network trained on the splits described in the previous section.

KITTI Odometry. The KITTI Odometry Benchmark contains 11 (0-10) sequences with provided ground-truth poses. As summarized in [78], sequences 00, 03, 04, 05, 07 are in the training set of the Eigen split that the proposed network uses, so we consider the rest of the sequences as the testing set for evaluating the pose estimation of D3VO. We use the relative translational (t_{rel}) error proposed in [25] as the main metric for evaluation. Table 4 shows the comparison with other state-of-the-art *mono* (M) as well as *stereo* (S) VO methods on the rest of the sequences. We refer to [78] for the results of the compared methods. Traditional monocular methods show high errors in the large-scale outdoor scene like the sequences in KITTI due to the scale drift. D3VO achieves the best performance on average, despite being a monocular methods as well. The table also contains the ablation study on the integration of deep depth (*Dd*), pose (*Dp*) and uncertainty (*Du*). It can be noticed that, consistent with the results in Table 1, the predicted uncertainty helps a lot on KITTI. We also submit the results on the testing sequences (11-20) to the KITTI Odometry evaluation server (link). At the time of submission, D3VO outperforms DVSO and achieves the best monocular VO performance and comparable to other state-of-the-art LiDAR and stereo methods.

We further compare D3VO with state-of-the-art end-to-end deep learning methods and other recent hybrid methods and show the results in Table 5. Note that here we only show the results on Seq.09 and 10, since most of the end-to-end methods only provide the results on these two sequences.

		01	02	06	08	09	10	mean
M	DSO [16]	9.17	114	42.2	177	28.1	24.0	65.8
	ORB [52]	108	10.3	14.6	11.5	9.30	2.57	37.0
S	S. LSD [18]	2.13	1.09	1.28	1.24	1.22	0.75	1.29
	ORB2 [53]	1.38	0.81	0.82	1.07	0.82	*0.58*	0.91
	S. DSO [74]	1.43	**0.78**	*0.67*	**0.98**	0.98	**0.49**	0.89
	Dd	1.16	0.84	0.71	1.01	0.82	0.73	0.88
	Dd+Dp	1.15	0.84	0.70	1.03	*0.80*	0.72	0.87
	Dd+Du	*1.10*	0.81	*0.69*	1.03	**0.78**	*0.62*	*0.84*
	D3VO	**1.07**	*0.80*	**0.67**	*1.00*	*0.78*	*0.62*	**0.82**

Table 4: Results on our test split of KITTI Odometry. The results of the SOTA monocular (M) methods are shown as baselines. The comparison with the SOTA stereo (S) methods shows that D3VO achieves better average performance than other methods, while being a monocular VO. We also show the ablation study for the integration of deep depth(*Dd*), pose(*Dp*) as well as uncertainty(*Du*).

		Seq. 09	Seq. 10
End-to-end	UnDeepVO [46]	7.01	10.63
	SfMLearner [86]	17.84	37.91
	Zhan et al. [82]	11.92	12.45
	Struct2Depth [6]	10.2	28.9
	Bian et al. [1]	11.2	10.1
	SGANVO [21]	*4.95*	**5.89**
	Gordon et al. [28]	**2.7**	*6.8*
Hybrid	CNN-SVO [48]	10.69	4.84
	Yin et al. [80]	4.14	1.70
	Zhan et al. [83]	2.61	2.29
	DVSO [78]	*0.83*	*0.74*
	D3VO	**0.78**	**0.62**

Table 5: Comparison to other hybrid methods as well as end-to-end methods on Seq.09 and 10 of KITTI Odometry.

We refer to [28, 78, 83] for the results for the compared methods. D3VO achieves better performance than all the end-to-end methods by a notable margin. In general, hybrid methods which combine deep learning with traditional methods deliver better results than end-to-end methods.

EuRoC MAV. As introduced in Sec. 4.1, EuRoC MAV is very challenging for purely vision-based VO due to the strong motion and significant illumination changes. VIO methods [44, 56, 71, 72] dominate this benchmark by integrating IMU measurements to get a pose or motion prior and meanwhile estimating the absolute scale. We compare D3VO with other state-of-the-art monocular VIO (M+I) as well as stereo VIO (S+I) methods on sequences *MH_03_medium*, *MH_05_difficult*, *V1_03_difficult*, *V2_02_medium* and *V2_03_difficult*. All the other sequences are used for training. We refer to [9] for the results of the M+I methods. The results of DSO and ORB-SLAM are shown as baselines. We also show the results from the proposed PoseNet *(End-end VO)*. For the evaluation metric, we use the root mean square (RMS) of the absolute trajectory error (ATE) after aligning the estimates with ground truth. The results in Table 6 show that with the proposed framework integrating depth, pose and uncertainty from the pro-

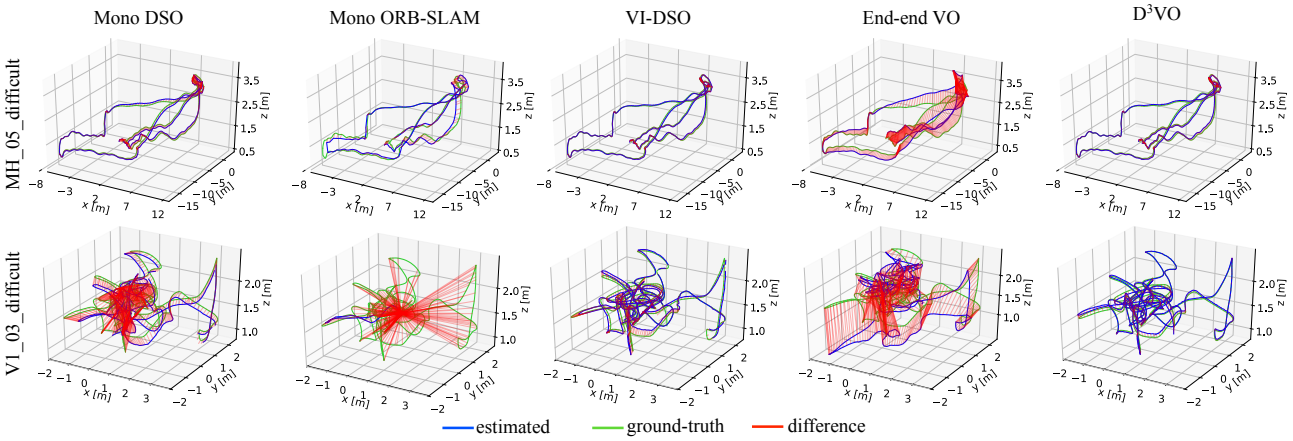


Figure 5: Qualitative comparison of the trajectories on *MH_05_difficult* and *V1_03_difficult* from EuRoC MAV.

		M03	M05	V103	V202	V203	mean
M	DSO [16]	0.18	0.11	1.42	0.12	0.56	0.48
M	ORB [52]	**0.08**	0.16	1.48	1.72	0.17	0.72
M+I	VINS [57]	0.13	0.35	0.13	0.08	0.21	0.18
M+I	OKVIS [44]	0.24	0.47	0.24	0.16	0.29	0.28
M+I	ROVIO [3]	0.25	0.52	0.14	0.14	0.14	0.24
M+I	MSCKF [51]	0.23	0.48	0.24	0.16	*0.13*	0.25
M+I	SVO [22]	0.12	0.16	X	X	X	0.14+X
M+I	VI-ORB [54]	*0.09*	**0.08**	X	**0.04**	**0.07**	0.07+X
M+I	VI-DSO [72]	0.12	0.12	**0.10**	0.06	*0.17*	*0.11*
M+I	*End-end VO*	1.80	0.88	1.00	1.24	0.78	1.14
M+I	*Dd*	0.12	0.11	0.63	0.07	0.52	0.29
M+I	*Dd+Dp*	*0.09*	*0.09*	0.13	0.06	0.19	0.11
M+I	*Dd+Du*	**0.08**	*0.09*	0.55	0.08	0.47	0.25
M+I	D3VO	**0.08**	*0.09*	*0.11*	*0.05*	0.19	**0.10**
S+I	VINS [57]	0.23	0.19	*0.11*	*0.10*	-	*0.17*
S+I	OKVIS [44]	0.23	0.36	0.13	0.17	-	0.22
S+I	Basalt [71]	**0.06**	*0.12*	**0.10**	**0.05**	-	**0.08**
S+I	D3VO	*0.08*	**0.09**	*0.11*	**0.05**	-	**0.08**

Table 6: Evaluation results on EuRoC MAV [5]. We show the results of DSO and ORB-SLAM as baselines and compare D3VO with other SOTA monocular VIO (M+I) and stereo VIO (S+I) methods. Note that for stereo methods, *V2_03_difficult* is excluded due to many missing images from one of the cameras [71]. Despite being a monocular method, D3VO shows comparable results to SOTA monocular/stereo VIO. The best results among the monocular methods are shown as **black bold** and the best among the stereo methods are shown as **blue bold**. The ablation study shows that *Dd+Dp* delivers large improvement on *V1_03_difficult* and *V2_03_difficult* where the camera motions are very strong.

posed deep neural network, D3VO shows high accuracy as well as robustness and is able to deliver comparable results to other state-of-the-art VIO methods with only a single camera. We also show the ablation study for the integration of predicted depth (*Dd*), pose (*Dp*) and uncertainty (*Du*) and the integration of pose prediction improves the performance significantly on *V1_03_difficult* and *V2_03_difficult* where violent camera motion occurs.

Figure 5 shows the qualitative comparison of trajectories obtained from DSO [16], ORB-SLAM [52], visual inertial DSO [72], the end-to-end predicted poses from our network and D3VO on the *MH_03* and *V1_03* sequences. All the 5 methods can deliver fairly good results on *MH_05_difficult*. On *V1_03_difficult* where the motions are stronger and there are many brightness inconsistencies between temporal and stereo images, D3VO can still deliver comparable results to VI-DSO, while using only a single camera.

5. Conclusion

We presented D3VO as a monocular VO method that enhances the performance of geometric VO methods by exploiting the predictive power of deep networks on three levels integrating predictions of monocular depth, photometric uncertainty and relative camera pose. To this end, we first introduced a novel self-supervised monocular depth estimation network which explicitly addresses the illumination change in the training set with predictive brightness transformation parameters. The network achieves state-of-the-art results on KITTI and EuRoC MAV. The predicted depth, uncertainty and pose are then incorporated into both the front-end tracking and back-end non-linear optimization of a direct VO pipeline. We systematically evaluated the VO performance of D3VO on the two datasets. D3VO sets a new state-of-the-art on KITTI Odometry and also achieves state-of-the-art performance on the challenging EuRoC MAV, rivaling with leading mono-inertial and stereo-inertial methods while using only a single camera.

Acknowledgements We would like to thank Niclas Zeller, Lukas Köstler, Oleg Muratov and other colleagues from Artisense for their continuous feedbacks. Besides, we would like to thank Jakob Engel and Tao Wu for the fruitful discussions during the early stages of the project. Last but not least, we also would like to thank the reviewers and Klaus H. Strobl for their constructive comments.

References

[1] 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 *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 7

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM-learning a compact, optimisable representation for dense visual SLAM. *arXiv preprint arXiv:1804.00874*, 2018. 3

[3] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015. 8

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 14

[5] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 2, 4, 5, 6, 8, 13

[6] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8001–8008, 2019. 7

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 13

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6, 15

[9] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018. 7

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*, 2018. 3

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 1

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 1

[13] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A trainable CNN for joint detection and description of local features. 2019. 1

[14] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 2

[15] 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*, pages 2366–2374, 2014. 2, 6

[16] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 2017. 1, 2, 3, 4, 5, 7, 8

[17] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014. 1, 3, 5

[18] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct SLAM with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE, 2015. 4, 7, 13

[19] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2, 5

[20] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003. 14

[21] Tuo Feng and Dongbing Gu. SGANVO: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks. *IEEE Robotics and Automation Letters*, 4(4):4431–4437, 2019. 7

[22] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016. 8

[23] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014. 5

[24] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, United States, 2018. 2

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 4, 5, 7, 13, 14

[26] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 6, 7, 13

[27] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-

right consistency. *arXiv preprint arXiv:1609.03677*, 2016. 1, 2, 3, 4, 6

[28] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2, 6, 7, 14

[29] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *Robotics and automation (ICRA), 2014 IEEE international conference on*, pages 1524–1531. IEEE, 2014. 3

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 13

[32] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2, 3

[33] Hailin Jin, Paolo Favaro, and Stefano Soatto. Real-time feature tracking and outlier rejection with changes in illumination. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 684–689. IEEE, 2001. 4

[34] E. Jung, N. Yang, and D. Cremers. Multi-Frame GAN: Image Enhancement for Stereo Visual Odometry in Low Light. In *Conference on Robot Learning (CoRL)*, 2019. 3

[35] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. 2, 3, 4

[36] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[37] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual SLAM for RGB-D cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013. 2

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 13

[39] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 1

[40] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning SFM from SFM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–713, 2018. 2, 3, 4, 5

[41] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001. 5

[42] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. *arXiv preprint arXiv:1702.02706*, 2017. 1, 6

[43] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. 1, 2

[44] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. 1, 7, 8

[45] Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015. 2

[46] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular visual odometry through unsupervised deep learning. *arXiv preprint arXiv:1709.06841*, 2017. 3, 7

[47] H-A Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004. 5

[48] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5218–5223. IEEE, 2019. 1, 7

[49] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, June 2018. 2

[50] Agostino Martinelli. Closed-form solution of visual-inertial structure from motion. *International journal of computer vision*, 106(2):138–152, 2014. 2

[51] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007. 8

[52] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1, 7, 8

[53] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 1, 3, 7, 14

[54] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017. 1, 8

[55] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic dif-

ferentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 13

[56] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 1, 7

[57] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019. 8

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4

[60] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 13

[61] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019. 2

[62] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular SLAM. *Robotics: Science and Systems VI*, 2, 2010. 1

[63] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664, May 2010. 2

[64] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012. 3

[65] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8934–8943, 2018. 1

[66] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006. 2, 5

[67] Jiexiong Tang, Ludvig Ericson, John Folkesson, and Patric Jensfelt. GCNv2: Efficient correspondence prediction for real-time slam. *IEEE Robotics and Automation Letters*, 4(4):3505–3512, 2019. 3

[68] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. *arXiv preprint arXiv:1704.03489*, 2017. 1, 3

[69] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 2

[70] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017. 3

[71] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *arXiv preprint arXiv:1904.06504*, 2019. 7, 8

[72] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517. IEEE, 2018. 1, 2, 7, 8

[73] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, June 2018. 2, 3

[74] R. Wang, M. Schwörer, and D. Cremers. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017. 1, 2, 3, 4, 6, 7, 14

[75] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017. 3

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

[77] N. Yang, R. Wang, X. Gao, and D. Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias and rolling shutter effect. *IEEE Robotics and Automation Letters (RA-L)*, 3:2878–2885, Oct 2018. 1

[78] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833, 2018. 1, 2, 3, 5, 6, 7

[79] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016. 1

[80] Xiaochuan Yin, Xiangwei Wang, Xiaoguo Du, and Qijun Chen. Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5870–5878, 2017. 1, 7

[81] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 2

[82] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 340–349, June 2018. 2, 3, 7

[83] Huangying Zhan, Chamara Saroj Weerasekera, Jiawang Bian, and Ian Reid. Visual odometry revisited: What should be learnt? *arXiv preprint arXiv:1909.09803*, 2019. 1, 7

[84] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018. 14

[85] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTAM: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018. 3

[86] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017. 1, 2, 3, 6, 7, 13

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Qadeer Khan[1,2] Daniel Cremers[1,2]

***Abstract*— Direct SLAM methods have shown exceptional performance on odometry tasks. However, they are susceptible to dynamic lighting and weather changes while also suffering from a bad initialization on large baselines. To overcome this, we propose GN-Net: a network optimized with the novel Gauss-Newton loss for training weather invariant deep features, tailored for direct image alignment. Our network can be trained with pixel correspondences between images taken from different sequences. Experiments on both simulated and real-world datasets demonstrate that our approach is more robust against bad initialization, variations in day-time, and weather changes thereby outperforming state-of-the-art direct and indirect methods. Furthermore, we release an evaluation benchmark for relocalization tracking against different types of weather. Our benchmark is available at `https://vision.in.tum.de/gn-net`.**

I. Introduction

In recent years, very powerful visual SLAM algorithms have been proposed [1], [2]. In particular, direct visual SLAM methods have shown great performance, outperforming indirect methods on most benchmarks [3], [4], [5]. They directly leverage the brightness data of the sensor to estimate localization and 3D maps rather than extracting a heuristically selected sparse subset of feature points. As a result, they exhibit a boost in precision and robustness. Nevertheless, compared to indirect methods, direct methods suffer from two major drawbacks:

1) Direct methods need a good initialization, making them less robust for large baseline tracking or cameras with a low frame rate.
2) Direct methods cannot handle changing lighting/weather conditions. In such situations, their advantage of being able to pick up very subtle brightness variations becomes a disadvantage to the more lighting invariant features.

In the last years, researchers have tackled the multiple-daytime tracking challenge with deep learning approaches that are designed to convert nighttime images to daytime images *e.g.* using GANs [6], [7], [8]. While this improves the robustness to changing lighting, one may ask why images should be the best input representation. Could there be better alternate representations?

This paper addresses the problem of adapting direct SLAM methods to challenging lighting and weather conditions. In this work, we show how to convert images into a multi-dimensional feature map which is invariant to lighting/weather changes and has by construction a larger basin of convergence. Thereby we overcome the aforementioned problems simultaneously. The deep features are trained with a novel Gauss-Newton loss formulation in a self-supervised manner. We employ a Siamese network trained with labels obtained either from simulation data or any state-of-the-art SLAM algorithm. This eliminates the additional cost of human labeling that is typically necessary for training a neural network. We exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm for direct image alignment. For this, we propose the Gauss-Newton loss which is designed to maximize the probability of identifying the correct pixel correspondence. The proposed loss function thereby enforces a feature representation that is designed to admit a large basin of convergence for the subsequent Gauss-Newton optimization. The superiority of our method stems from its ability to generate these multi-channel, weather-invariant deep features that facilitate relocalization across different weathers. Figure 1 shows how our method can successfully relocalize a snowy sequence in a pre-built map created using a sunny sequence.

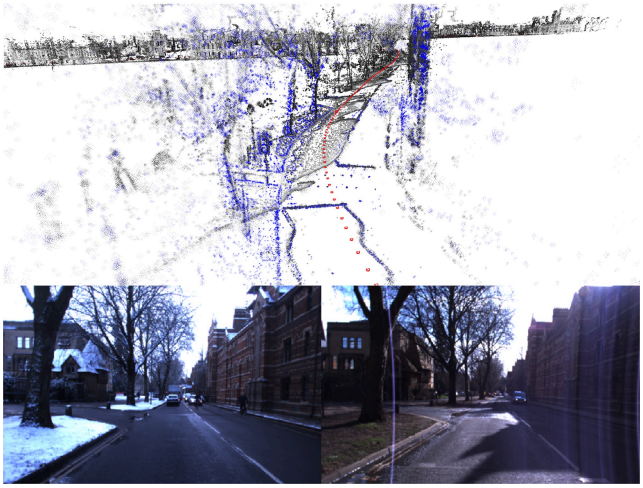


Fig. 1: We relocalize a snowy sequence from the Oxford RobotCar dataset in a pre-built map created using a sunny weather condition. The points from the prior map (gray) well align with the new points from the current run (blue), indicating that the relocalization is indeed accurate.

In common benchmarks [9], localizing accurately in a pre-built map has been tackled by finding nearby images (*e.g.* by using NetVLAD [10]) and tracking the relative pose (6DOF) between them. However, we propose to split this into two separate tasks. In this work, we focus on the second challenge which we refer to as *relocalization tracking*. This way, we

*These authors contributed equally.
[1]Technical University of Munich
[2]Artisense

can evaluate its performance in isolation. This is formalized to what we refer to as *relocalization tracking*. Since there is no publicly available dataset to evaluate *relocalization tracking* performance across multiple types of weathers, we are releasing an evaluation benchmark having the following 3 attributes:

- It contains sequences from multiple different kinds of weathers.
- Pixel-wise correspondences between sequences are provided for both simulated and real-world datasets.
- It decouples relocalization tracking from the image retrieval task.

The challenge here in comparison with normal pose estimation datasets [11], [12] is that the images involved are usually captured at different daytimes/seasons and there is no good initialization of the pose. We summarize the main contributions of our paper as:

- We derive the Gauss-Newton loss formulation based on the properties of direct image alignment and demonstrate that it improves the robustness to large baselines and illumination/weather changes.
- Our experimental evaluation shows, that GN-Net outperforms both state-of-the-art direct and indirect SLAM methods on the task of *relocalization tracking*.
- We release a new evaluation benchmark for the task of *relocalization tracking* with ground-truth poses. It is collected under dynamic conditions such as illumination changes, and different weathers. Sequences are taken from the the CARLA [13] simulator as well as from the Oxford RobotCar dataset [14].

II. Related Work

We review the following main areas of related work: visual SLAM, visual descriptor learning, deep direct image alignment, and image-based relocalization in SLAM.

Direct versus indirect SLAM methods: Most existing SLAM systems that have used feature descriptors are based on traditional manual feature engineering, such as ORB-SLAM [2], MonoSLAM [15], and PTAM [1].

An alternative to feature-based methods is to skip the pre-processing step of the raw sensor measurements and rather use the pixel intensities directly. Popular direct visual methods are DTAM [16], LSD-SLAM [3], DSO [5], and PhotoBundle [4]. However, the main limitation of direct methods is the *brightness constancy* assumption which is rarely fulfilled in any real-world robotic application [17]. The authors of [18] propose to use binary feature descriptors for direct tracking called Bit-planes. While improving the robustness to bad lighting situations it was also found that Bit-planes have a smaller convergence basin than intensities. This makes their method less robust to bad initialization. In contrast, the features we propose *both* improve robustness to lighting and the convergence basin.

Visual descriptor learning: Feature descriptors play an important role in a variety of computer vision tasks. For example, [19] proposed a novel correspondence contrastive loss which allows for faster training and demonstrates their effectiveness for both geometric and semantic matching across intra-class shape or appearance variations. In [20], a deep neural network is trained using a contrastive loss to produce viewpoint- and lighting-invariant descriptors for single-frame localization. The authors of [21] proposed a CNN-based model that learns local patterns for image matching without a global geometric model. [22] uses convolutional neural networks to compute descriptors which allow for efficient detection of poorly textured objects and estimation of their 3D pose. In [23], the authors propose to train features for optical flow estimation using a Hinge loss based on correspondences. In contrast to our work, their loss function does not have a probabilistic derivation and they do not apply their features to pose estimation. [24] uses deep learning to improve SLAM performance in challenging situations. They synthetically create images and choose the one with most gradient information as the ground-truth for training. In contrast to them, we do not limit our network to output images similar to the real world. In [25], the authors compare dense descriptors from a standard CNN, SIFT, and normal image intensities for dense Lucas-Kanade tracking. There, it can be seen that grayscale values have a better convergence basin than the other features, which is something we overcome with our approach.

Deep direct image alignment: BA-NET [26] introduces a network architecture to solve the structure from motion (SfM) problem via feature-metric bundle adjustment. Unlike the BA-NET, instead of predicting the depth and the camera motion simultaneously, we propose to only train on correspondences obtained from a direct SLAM system. The advantage is that correspondences are oftentimes easier to obtain than accurate ground-truth poses. Furthermore, we combine our method with a state-of-the-art direct SLAM system and utilize its depth estimation, whereas BA-NET purely relies on deep learning. RegNet [27] is another line of work which tries to replace the handcrafted numerical Jacobian by a learned Jacobian with the help of a depth prediction neural network. However, predicting a dense depth map is often inaccurate and computationally demanding. The authors of [28] propose to use a learning-based inverse compositional algorithm for dense image alignment. The drawback of this approach is that the algorithm is very sensitive to the data distribution and constrained towards selecting the right hyperparameters. In [29] they use high-dimensional features in a direct image alignment framework for monocular VO. In contrast to us, they only use already existing features and do not apply them for relocalization.

Relocalization: An important task of relocalization is to approximate the pose of an image by simply querying the most similar image from a database [30], [31]. However, this has only limited accuracy unless the 6DOF pose between the queried and the current image is estimated in a second step. Typically, this works by matching 2D-3D correspondences between an image and a point cloud and estimating the pose using indirect image alignment [32]. In contrast, we propose to use direct image alignment paired with deep features.

Relocalization benchmarks: The authors of [9] have done sequence alignment on the Oxford RobotCar dataset, however, they have not made the matching correspondences public. The Photo Tourism [33] is another dataset providing images and ground-truth correspondences of popular monuments from different camera angles and across different weather/lighting conditions. However, since the images are not recorded as a sequence, relocalization tracking is not possible. Furthermore, their benchmark only supports the submission of features rather than poses, thereby restricting evaluation to only indirect methods.

III. Deep Direct SLAM

In this work, we argue that a network trained to output features which produce better inputs for direct SLAM as opposed to normal images should have the following properties:

- Pixels corresponding to the same 3D point should have similar features.
- Pixels corresponding to different 3D points should have dissimilar features.
- When starting in a vicinity around the correct pixel, the Gauss-Newton algorithm should move towards the correct solution.

For optimizing the last property, we propose the novel Gauss-Newton loss which makes use of the probabilistic background of the Gauss-Newton algorithm for direct image alignment. The final loss is a weighted sum of the pixel-wise contrastive loss and the Gauss-Newton loss.

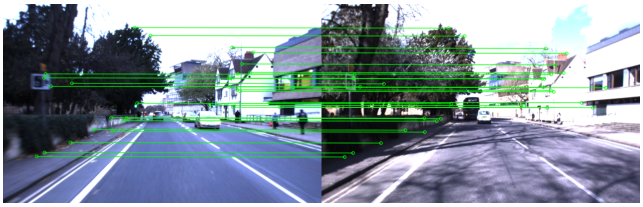


Fig. 2: This figure shows training correspondences between a pair of images from our benchmark.

Architecture: We are interested in learning a non-linear mapping, which maps images, $\mathbb{R}^{W\times H\times C}$ to a dense visual descriptor space, $\mathbb{R}^{W\times H\times D}$, where each pixel is represented by a D-dimensional vector. The training is performed by a Siamese encoder-decoder structured network, where we feed a pair of images, $\mathbf{I}_a$ and $\mathbf{I}_b$, producing multi-scale feature pyramids $\mathbf{F}_a^l$ and $\mathbf{F}_b^l$, where l represents the level of the decoder. For each image pair, we use a certain number of matches, denoted by N_{pos}, and a certain number of non-matches, denoted by N_{neg}. A pixel $\mathbf{u}_a \in \mathbb{R}^2$ from image $\mathbf{I}_a$ is considered to be a positive example if the pixel $\mathbf{u}_b \in \mathbb{R}^2$ from image $\mathbf{I}_b$ corresponds to the same 3D vertex (Figure 2). We make use of the inherent multi-scale hierarchy of the U-Net [34] architecture to apply the different loss terms from coarser to finer scaled pyramid levels. With this approach, our learned features will have a larger convergence radius for visual SLAM methods.

Pixelwise contrastive loss: The pixelwise contrastive loss attempts to minimize the distance between positive pairs, and maximize the distance between negative pairs. It can be computed as follows: $\mathcal{L}_{\text{contrastive}}(\mathbf{F}_a, \mathbf{F}_b, l) = \mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) + \mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l)$.

$$\mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{pos}}} \sum_{N_{\text{pos}}} D_{\text{feat}}^2 \quad (1)$$

$$\mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{neg}}} \sum_{N_{\text{neg}}} \max(0, M - D_{\text{feat}})^2 \quad (2)$$

where $D_{\text{feat}}(\cdot)$ is the L_2 distance between the feature embeddings: $D_{\text{feat}} = ||\mathbf{F}_a^l(\mathbf{u}_a) - \mathbf{F}_b^l(\mathbf{u}_b)||_2$ and M is the margin and set to 1.

Gauss-Newton algorithm for direct image alignment: Our learned deep features are ultimately applied to pose estimation. This is done using direct image alignment but generalized to a multi-channel feature map $\mathbf{F}$ with D channels. The input to this algorithm is a reference feature map $\mathbf{F}$ with known depths for some pixels in the image, and a target feature map $\mathbf{F}'$. The output is the predicted relative pose $\boldsymbol{\xi}$. Starting from an initial guess the following steps are performed iteratively:

1) All points $\mathbf{p}_i$ with known depth values are projected from the reference feature map $\mathbf{F}$ into the target feature map $\mathbf{F}'$ yielding the point $\mathbf{p'}_i$. For each of them a residual vector $\mathbf{r} \in \mathbb{R}^D$ is computed, enforcing that the reference pixel and the target pixel should be similar:
$$\mathbf{r}_i(\mathbf{p}_i, \mathbf{p'}_i) = \mathbf{F}'(\mathbf{p'}_i) - \mathbf{F}(\mathbf{p}_i) \quad (3)$$
2) For each residual the derivative with respect to the relative pose is:
$$\mathbf{J}_i = \frac{d\mathbf{r}_i}{d\boldsymbol{\xi}} = \frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i} \cdot \frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}} \quad (4)$$
Notice that the reference point $\mathbf{p}_i$ does not change for different solutions $\boldsymbol{\xi}$, therefore it does not appear in the derivative.
3) Using the stacked residual vector $\mathbf{r}$, the stacked Jacobian $\mathbf{J}$, and a diagonal weight matrix $\mathbf{W}$, the Gaussian system and the step $\boldsymbol{\delta}$ is computed as follows:
$$\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T\mathbf{W}\mathbf{r} \text{ and } \boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b} \quad (5)$$

Note that this derivation is equivalent to normal direct image alignment (as done in the frame-to-frame tracking from DSO) when replacing $\mathbf{F}$ with the image $\mathbf{I}$. In the computation of the Jacobian the numerical derivative of the features $\frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i}$ is used. As typical images are extremely non-convex this derivative is only valid in a small vicinity (usually 1-2 pixels) around the current solution which is the main reason why direct image alignment needs a good initialization. To partially overcome this, a pyramid scheme is often used. Usually tracking on multiple channels instead of one can decrease the convergence radius ([18], [25]). However, in our case, we train the feature maps to in fact

have a larger convergence basin than images by enforcing smoothness in the vicinity of the correct correspondence.

Gauss-Newton on individual pixels: Instead of running the Gauss-Newton algorithm on the 6DOF pose we can instead use it on each point $\mathbf{p}_i$ individually (which is similar to the Lucas-Kanade algorithm [35]). Compared to direct image alignment, this optimization problem has the same residual, but the parameter being optimized is the point position instead of the relative pose. In this case, the Hessian will be a 2-by-2 matrix and the step $\boldsymbol{\delta}$ can simply be added to the current pixel position (we leave out $\mathbf{W}$ for simplicity):

$$\mathbf{J'}_i = \frac{d\mathbf{F'}(\mathbf{p'}_i)}{d\mathbf{p'}_i} \text{ and } \mathbf{H'}_i = \mathbf{J'}_i^T \mathbf{J}_i \text{ and } \mathbf{b'}_i = \mathbf{J'}_i^T \mathbf{r}_i \quad (6)$$

These individual Gauss-Newton systems can be combined with the system for 6DOF pose estimation (Equation (5)) using:

$\mathbf{H} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{H'}_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)$ and $\mathbf{b} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{b'}_i$

The difference between our simplified systems and the one for pose estimation is only the derivative with respect to the pose, which is much smoother than the image derivative [5].

This means that if the Gauss-Newton algorithm performs well on individual pixels it will also work well on estimating the full pose. Therefore, we propose to train a neural network on correspondences which are easy to obtain, *e.g.* using a SLAM method, and then later apply it for pose estimation.

We argue that training on these individual points is superior to training on the 6DOF pose. The estimated pose can be correct even if some points contribute very wrong estimates. This increases robustness at runtime but when training we want to improve the information each point provides. Also, when training on the 6DOF pose we only have one supervision signal for each image pair, whereas when training on correspondences we have over a thousand signals. Hence, our method exhibits exceptional generalization capabilities as shown in the results section.

The probabilistic Gauss-Newton loss: The linear system described in Equation (6) defines a 2-dimensional Gaussian probability distribution. The reason is that the Gauss-Newton algorithm tries to find the solution with maximum probability in a least squares fashion. This can be derived using the negative log-likelihood of the Gaussian distribution:

$$E(\mathbf{x}) = -\log f_X(\mathbf{x}) = \quad (7)$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) = \quad (8)$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{H}(\mathbf{x}-\boldsymbol{\mu}) + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|) \quad (9)$$

where $\mathbf{x}$ is a pixel position and $\boldsymbol{\mu}$ is the mean.

In the Gauss-Newton algorithm the mean (which also corresponds to the point with maximum probability) is computed with $\boldsymbol{\mu} = \mathbf{x}_s + \boldsymbol{\delta}$, where the $\boldsymbol{\delta}$ comes from Equation (5) and $\mathbf{x}_s$ denotes the start point. To derive this, only the first term is used (because the latter parts are constant for all solutions $\mathbf{x}$). In our case, however, the second term is very relevant, because the network can influence both $\boldsymbol{\mu}$ and $\mathbf{H}$.

This derivation shows, that $\mathbf{H}, \mathbf{b}$ as computed in the GN-algorithm, also define a Gaussian probability distribution with mean $\mathbf{x}_s + \mathbf{H}^{-1}\mathbf{b}$ and covariance $\mathbf{H}^{-1}$.

When starting with an initial solution $\mathbf{x}_s$ the network should assign maximal probability to the pixel that marks the correct correspondence. With $\mathbf{x}$ being the correct correspondence, we therefore use $E(\mathbf{x})$ = Equation (9) as our loss function which we call the *Gauss-Newton loss* (see Algorithm 1).

Algorithm 1 Compute Gauss-Newton loss

$\mathbf{F}_a \leftarrow \text{network}(\mathbf{I}_a)$
$\mathbf{F}_b \leftarrow \text{network}(\mathbf{I}_b)$
$e \leftarrow 0$ ▷ Total error
for all correspondences $\mathbf{u}_a, \mathbf{u}_b$ **do**
  $\mathbf{f}_t \leftarrow \mathbf{F}_a(\mathbf{u}_a)$ ▷ Target feature
  $\mathbf{x}_s \leftarrow \mathbf{u}_b + \text{rand}(\text{vicinity})$ ▷ Compute start point
  $\mathbf{f}_s \leftarrow \mathbf{F}_b(\mathbf{x}_s)$
  $\mathbf{r} \leftarrow \mathbf{f}_s - \mathbf{f}_t$ ▷ Residual
  $\mathbf{J} \leftarrow \frac{d\mathbf{F}_b}{d\mathbf{x}_s}$ ▷ Numerical derivative
  $\mathbf{H} \leftarrow \mathbf{J}^T\mathbf{J} + \epsilon \cdot \text{Id}$ ▷ Added epsilon for invertibility
  $\mathbf{b} \leftarrow \mathbf{J}^T r$
  $\boldsymbol{\mu} \leftarrow \mathbf{x}_s - \mathbf{H}^{-1}\mathbf{b}$
  $e_1 \leftarrow \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$ ▷ First error term
  $e_2 \leftarrow \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$ ▷ Second error term
  $e \leftarrow e + e_1 + e_2$
end for

In the algorithm, a small number ϵ is added to the diagonal of the Hessian, to ensure it is invertible.

Analysis of the Gauss-Newton loss: By minimizing Equation (9) the network has to maximize the probability density of the correct solution. As the integral over the probability densities always has to be 1, the network has the choice to either focus all the density on a small set of solutions (with more risk of being penalized if this solution is wrong), or to distribute the density to more solutions which in turn will have a lower individual density. By maximizing the probability of the correct solution, the network is incentivized to improve the estimated solution and its certainty.

This is also reflected in the two parts of the loss. The first term $e_1 = \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$ penalizes deviations between the estimated and the correct solution, scaled with the Hessian $\mathbf{H}$. The second term $e_2 = \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$ is large if the network does not output enough certainty for its solution. This means that the network can reduce the first error term e_1 by making $\mathbf{H}$ smaller. As a consequence, the second error term will be increased, as this will also reduce the determinant of $\mathbf{H}$. Notice also that this can be done in both dimensions independently. The network has the ability to output a large uncertainty in one direction, but a small uncertainty in the other direction. This is one of the traditional advantages of direct methods which are naturally able to utilize also lines instead of just feature points.

From Equation (9) it can be observed that the predicted uncertainty depends only on the numerical derivative of the

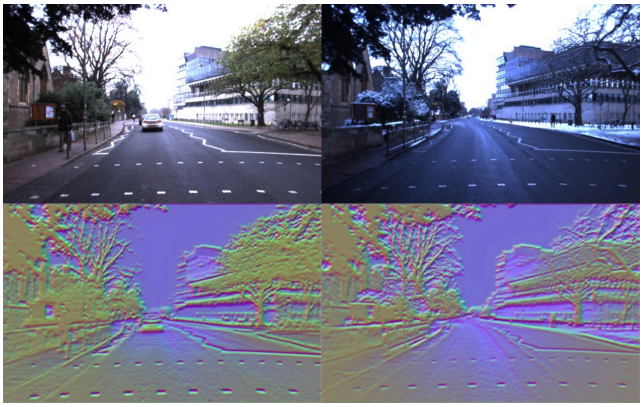


Fig. 3: This figure shows images and their corresponding feature maps predicted by our GN-Net for the Oxford RobotCar dataset. Each column depicts the image and feature map for a sample taken from 2 different sequences. Despite lighting and weather changes, the feature maps are robust to these variations. The visualization of the features shows the high-dimensional descriptors reduced to 3D through PCA.

target image at the start position. The higher the gradients the higher the predicted certainty. In DSO this is an unwanted effect that is counteracted by the gradient-dependent weighting applied to the cost-function [5, Equation (7)]. In our case, however, it gives the network the possibility to express its certainty and incentivizes it to output discriminative features.

Upon training the network with our loss formulation, we observe that the features are very similar despite being generated from images taken from sequences with different lighting/weather conditions, as shown in Figure 3.

IV. Relocalization Tracking Benchmark

Previous tasks for localization/odometry can primarily be divided into two categories:

- Odometry datasets [11], [12], where there is a continuous stream of images (sometimes combined with additional sensor data like IMUs).
- Image collections where individual images are usually further apart from each other in space/time [36], [9].

We argue that for several applications a combination of these two tasks which we refer to as *relocalization tracking* is a more realistic scenario. The idea is that the algorithm has two inputs:

1) An image sequence (like a normal odometry dataset).
2) A collection of individual images (possibly with different weathers/times), each of which shall be tracked against one specific image from point 1.

The algorithm is supposed to track the normal sequential image sequence and at the same time perform tracking of the images in point 2. The advantage of this task is that the used algorithm can utilize the temporally continuous sequence from point 1 to compute accurate depth values for a part of the image (using a standard visual odometry method), which can then be used to improve the tracking of the individual images of point 2.

This task is very realistic as it comes up when tracking an image sequence and at the same time trying to relocalize this sequence in a prior map. A similar challenge occurs by trying to merge multiple maps from different times. In both cases, one has more information than just a random collection of images. It is important to reiterate here that the task of finding relocalization candidates is not considered but rather tracking them with maximum accuracy/robustness is the focus. This is because our benchmark decouples image retrieval from tracking.

We have created a benchmark for *relocalization tracking* using the CARLA simulator and the Oxford RobotCar dataset. Our benchmark includes ground-truth poses between different sequences for both training, validation, and testing.

CARLA: For synthetic evaluations, we use CARLA version 0.8.2. We collect data for 3 different weather conditions representing *WetNoon*, *SoftRainNoon*, and *WetCloudySunset*. We recorded the images at a fixed framerate of 10 frames per second (FPS). At each time step, we record images and its corresponding dense depth map from 6 different cameras with different poses rendered from the simulation engine, which means that the poses in the benchmark are not limited to just 2DOF. The images and the dense depth maps are of size 512×512. For each weather condition, we collected 3 different sequences comprising 500-time steps with an average distance of 1.6m. This is done for training, validation, and testing, meaning there are 27 sequences, containing 6 cameras each. Training, validation, and test sequences were all recorded in different parts of the CARLA town. We have generated the test sequences after all hyperparameter tuning of our method was finished, meaning we had no access to the test data when developing the method. In accordance, we shall withhold the ground-truth for the test sequences.

Oxford RobotCar: Creating a multi-weather benchmark for this dataset imposes various challenges because the GPS-based ground-truth is very inaccurate. To find the relative poses between images from different sequences we have used the following approach. For pairs of images from two different sequences, we accumulate the point cloud captured by the 2D lidar for 60 meters using the visual odometry result provided by the Oxford dataset. The resulting two point clouds are aligned with the global registration followed by ICP alignment using the implementation of Open3D [37]. We provide the first pair of images manually and the following pairs are found using the previous solution. We have performed this alignment for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*. The average relocalization distance across all sequences is 0.84m.

V. Experimental Evaluation

We perform our experiments on the *relocalization tracking* benchmark described in Section IV. We demonstrate the multi-weather relocalization performance on both the

CARLA and the Oxford RobotCar dataset. For the latter, we show that our method even generalizes well to unseen weather conditions like rain or snow while being trained only on the sunny and overcast conditions. Furthermore, a qualitative relocalization demo[1] on the Oxford RobotCar dataset is provided, where we demonstrate that our GN-Net can facilitate precise relocalization between weather conditions.

We train our method using sparse depths created by running Stereo DSO on the training sequences. We use intra-sequence correspondences calculated using the DSO depths and the DSO pose. Meanwhile, inter-sequence correspondences are obtained using DSO depths and the ground-truth poses provided by our benchmark. The ground truth poses are obtained via Lidar alignment for Oxford and directly from the simulation engine for CARLA as explained in Section IV. Training is done from scratch with randomly initialized weights and an ADAM optimizer with a learning rate of 10^{-6}. The image pair fed to the Siamese network is randomly selected from any of the training sequences while ensuring that the images in the pair do not differ by more than 5 keyframes. Each branch of the Siamese network is a modified U-Net architecture with shared weights. Further details of the architecture and training can be found in the supplementary material[1]. Note that at inference time, only one image is needed to extract the deep visual descriptors, used as input to the SLAM algorithm. While in principle, our approach can be deployed in conjunction with any direct method, we have coupled our deep features with Direct Sparse Odometry (DSO).

We compare to state-of-the-art **direct** methods:

Stereo Direct Sparse Odometry (DSO) [38]: Whenever there is a relocalization candidate for a frame we ensure that the system creates the corresponding keyframe. This candidate is tracked using the *coarse tracker*, performing direct image alignment in a pyramid scheme. We use the identity as initialization without any other random guesses for the pose.

GN-Net (Ours): Same as with DSO, however, for relocalization tracking, we replace the grayscale images with features created by our GN-Net on all levels of the feature pyramid. The network is trained with the Gauss-Newton loss formulation described in Section III.

We also compare to state-of-the-art **indirect** methods:

ORB-SLAM [32]: For relocalization tracking, we use the standard feature-based 2-frame pose optimization also used for frame-to-keyframe tracking. We have also tried the RANSAC scheme implemented in ORB-SLAM for relocalization, however, it yielded worse results overall. Thus we will report only the default results.

D2-Net [39], SuperPoint [40]: For both methods we use the models provided by the authors. The relative pose is estimated using the OpenCV implementation of the PnP algorithm in a RANSAC scheme.

We also evaluated the Deeper Inverse Compositional Algorithm [28] on the *relocalization tracking* benchmark. However, the original implementation didn't converge despite multiple training trials with different hyperparameters.

For all our quantitative experiments we plot a cumulative distribution of the relocalization error, which is the norm of the translation between the estimated and the correct solution in meters. For each relocalization error between 0 and 1 meter, it plots the percentage of relocalization candidates that have been tracked with at least this accuracy.

A. Quantitative multi-weather evaluation

We demonstrate the relocalization tracking accuracy on our new benchmark across different weathers. For these experiments, tracking is performed only across sequences with a different weather condition.

CARLA: For this experiment, we train on the training sequences provided by our benchmark. For all learning-based approaches, the best epoch is selected using the relocalization tracking performance on the validation set. The results on the test data are shown in the supplementary[1].

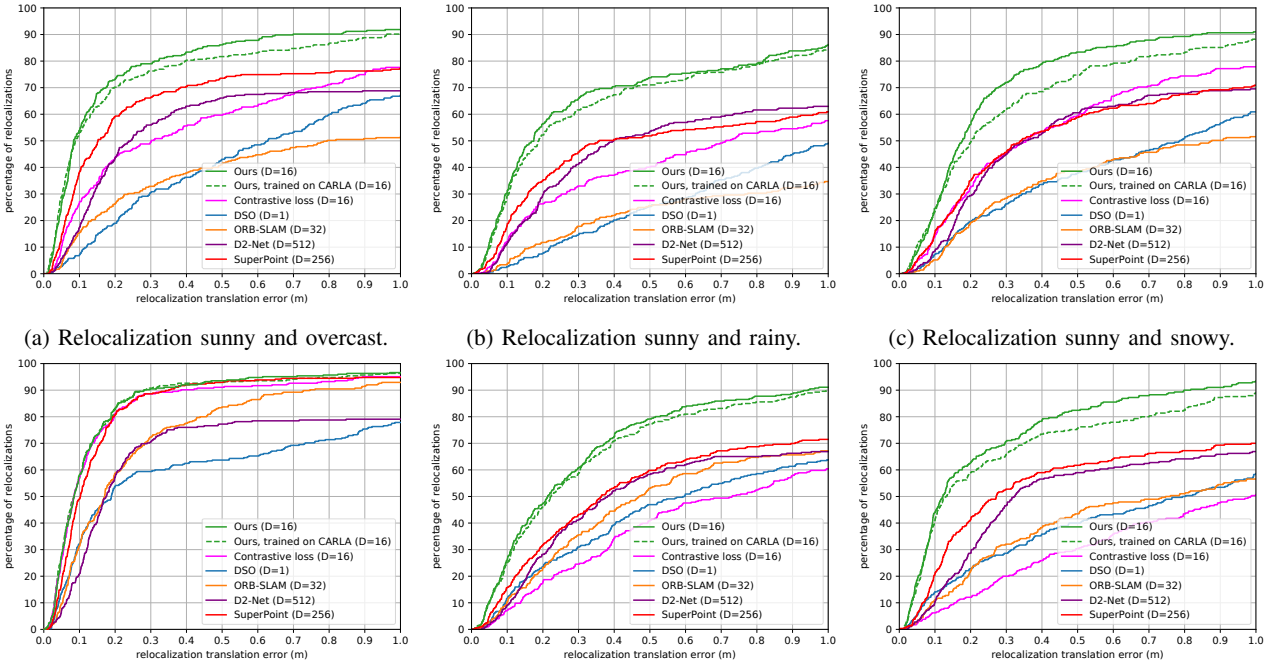
Oxford RobotCar: We train on the sunny and overcast condition correspondences provided by our *relocalization tracking* benchmark for the Oxford dataset. For the learning-based methods, we select the best epoch based on the relocalization tracking performance on the training set. We use the same hyperparameters that were found using the CARLA validation set. We show the results on the test data in Figure 4. Our method significantly outperforms the baselines. The Gauss-Newton loss has a large impact as compared to the model trained with only the contrastive loss.

Figures 4b-f show how well our model generalizes to unseen weather conditions. Despite being trained only on two sequences with overcast and sunny conditions the results for tracking against a rainy and a snowy sequence are almost the same. Interestingly our model which was trained only on the CARLA benchmark outperforms all baselines significantly.

B. Qualitative multi-weather evaluation

Finally, we show a relocalization demo comparing our GN-Net to DSO. For this, we load a point cloud from a sequence recorded in the sunny condition and relocalize against sequences from rainy and snowy conditions. For each keyframe, we try to track it against the nearest keyframe in the map according to the currently estimated transformation between the trajectory and the map. Figure 6 shows that the point clouds from the different sequences align nicely, despite belonging to different weather conditions. This experiment shows that our method can perform the desired operations successfully on a real-world application, including relocalization against unseen weather conditions. Figure 7 demonstrates the difference between our Gauss-Newton loss and the contrastive loss. This shows that the quantitative improvement has a visible effect on the application of relocalization. Figure 5 shows sample images used in the qualitative relocalizations.

[1] https://vision.in.tum.de/gn-net.



(a) Relocalization sunny and overcast. (b) Relocalization sunny and rainy. (c) Relocalization sunny and snowy.

(d) Relocalization overcast and rainy. (e) Relocalization overcast and snowy. (f) Relocalization rainy and snowy.

Fig. 4: This figure shows the cumulative relocalization accuracy on the Oxford RobotCar dataset for different sequences. D denotes the dimension of the feature descriptor. Our method achieves the highest accuracy across all sequences. It is interesting to observe that despite being trained only on two sequences in overcast and sunny condition, our model still generalizes very well to even *unseen* rainy and snowy conditions. Even the model trained only on the synthetic CARLA benchmark outperforms all baselines, showing exceptional generalization capabilities.

Fig. 5: shows image pairs used in the qualitative relocalizations. Left: rainy (top row) and snowy (bottom row) images relocalized against the sunny reference images (right).

C. Additional experiments on EuRoC and CARLA

In the supplementary, we provide more evaluations on datasets with and without brightness variations. This includes relocalization tracking on the CARLA benchmark and visual odometry on the EuRoC [11] dataset. We show that also in these situations our deep features significantly outperform DSO and ORB-SLAM because of their robustness to large-baselines. On the EuRoC dataset, we improve the DSO performance by almost a factor of 2 for low-framerates.

VI. Conclusion & Future Work

With the advent of deep learning, we can devise feature space encodings that are designed to be optimally suited for the subsequent visual SLAM algorithms. More specifically, we propose to exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm to devise a novel loss function for feature space encoding that we call the Gauss-Newton loss. It is designed to promote robustness to strong lighting and weather changes while enforcing a maximal basin of convergence for the respective SLAM algorithm. Quantitative experiments on synthetic and real-world data demonstrates that the Gauss-Newton loss allows us to significantly expand the realm of applicability of direct visual SLAM methods, enabling relocalization and map merging across drastic variations in weather and illumination.

References

[1] G. Klein and D. W. Murray, "Parallel tracking and mapping for small ar workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[2] R. Mur-Artal, J. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE T-RO*, vol. 31, no. 5, 2015.

[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[4] H. Alismail, B. Browning, and S. Lucey, "Photometric Bundle Adjustment for Vision-Based SLAM," in *ACCV*, 2017.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE TPAMI*, vol. 40, no. 3, 2018.

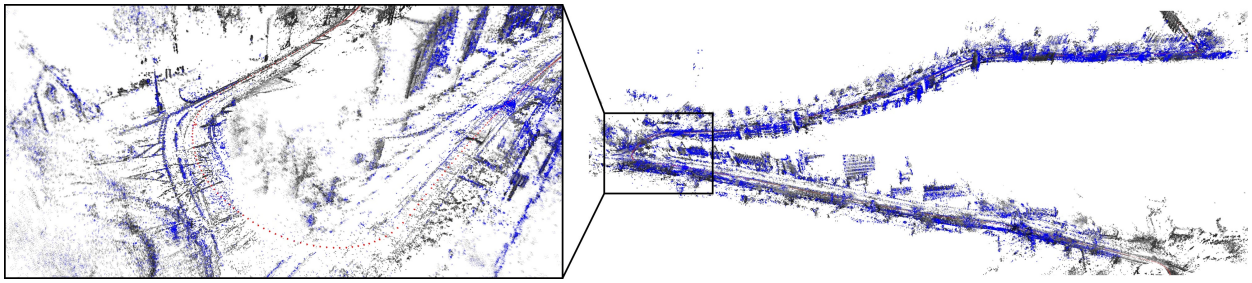


Fig. 6: This figure shows a point cloud result of our GN-Net. We relocalize a rainy sequence (blue) against a reference map created from the sunny sequence (gray).

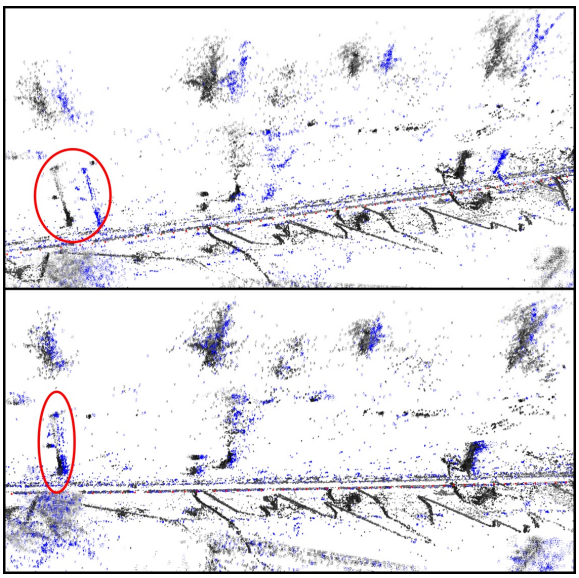


Fig. 7: Top: relocalization using the model trained with only the contrastive loss. Bottom: relocalization using the model trained with our loss formulation. This visually demonstrates the influence of the Gauss-Newton loss.

[6] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," in *NIPS*, 2017.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *IEEE CVPR*, 2017.

[8] H. Porav, W. Maddern, and P. Newman, "Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer," in *IEEE ICRA*, 2018.

[9] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions," in *IEEE CVPR*, 2018.

[10] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *IEEE CVPR*, 2016.

[11] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC Micro Aerial Vehicle Datasets," *IJRR*, vol. 35, no. 10, 2016.

[12] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," in *arXiv 2016*, 2016.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *CoRL*, 2017.

[14] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *IJRR*, vol. 36, no. 1, 2017.

[15] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE TPAMI*, no. 6, 2007.

[16] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *ICCV*, 2011.

[17] S. Park, T. Schöps, and M. Pollefeys, "Illumination Change Robustness in Direct Visual SLAM," in *IEEE ICRA*, 2017.

[18] H. Alismail, M. Kaess, B. Browning, and S. Lucey, "Direct Visual Odometry in Low Light Using Binary Descriptors," *RA-L*, vol. 2, 2017.

[19] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal Correspondence Network," in *NIPS*, 2016.

[20] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised Visual Descriptor Learning for Dense Correspondence," *RA-L*, vol. 2, 2017.

[21] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," in *NeurIPS*, 2018.

[22] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," *IEEE CVPR*, 2015.

[23] C.-H. Chang, C.-N. Chou, and E. Y. Chang, "CLKN: Cascaded Lucas-Kanade Networks for Image Alignment," in *IEEE CVPR*, 2017.

[24] R. Gomez-Ojeda, Z. Zhang, J. Gonzalez-Jimenez, and D. Scaramuzza, "Learning-based image enhancement for visual odometry in challenging hdr environments," in *IEEE ICRA*, 2018.

[25] J. Czarnowski, S. Leutenegger, and A. J. Davison, "Semantic Texture for Robust Dense Tracking," in *ICCVW*, 2017.

[26] C. Tang and P. Tan, "BA-Net: Dense Bundle Adjustment Network," in *ICLR*, 2019.

[27] L. Han, M. Ji, L. Fang, and M. Nießner, "Regnet: Learning the optimization of direct image-to-image pose registration," in *arXiv 2018*, 2018.

[28] Z. Lv, F. Dellaert, J. Rehg, and A. Geiger, "Taking a deeper look at the inverse compositional algorithm," in *IEEE CVPR*, 2019.

[29] C. Jaramillo, Y. Taguchi, and C. Feng, "Direct multichannel tracking," in *3DV*, 2017.

[30] M. Cummins and P. Newman, "FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance," *IJRR*, vol. 27, no. 6, 2008.

[31] I. Kapsouras and N. Nikolaidis, "A vector of locally aggregated descriptors framework for action recognition on motion capture data," in *EUSIPCO*, 2018.

[32] R. Mur-Artal and J. D. Tards, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE T-RO*, vol. 33, no. 5, 2017.

[33] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH*, 2006.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.

[35] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, 1981.

[36] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *ICCV*, 2015.

[37] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv 2018*, 2018.

[38] R. Wang, M. Schwörer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *ICCV*, 2017.

[39] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features," in *IEEE CVPR*, 2019.

[40] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," in *IEEE CVPRW*, 2018.

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Nan Yang[1,2] Daniel Cremers[1,2]
[1] Technical University of Munich [2] Artisense

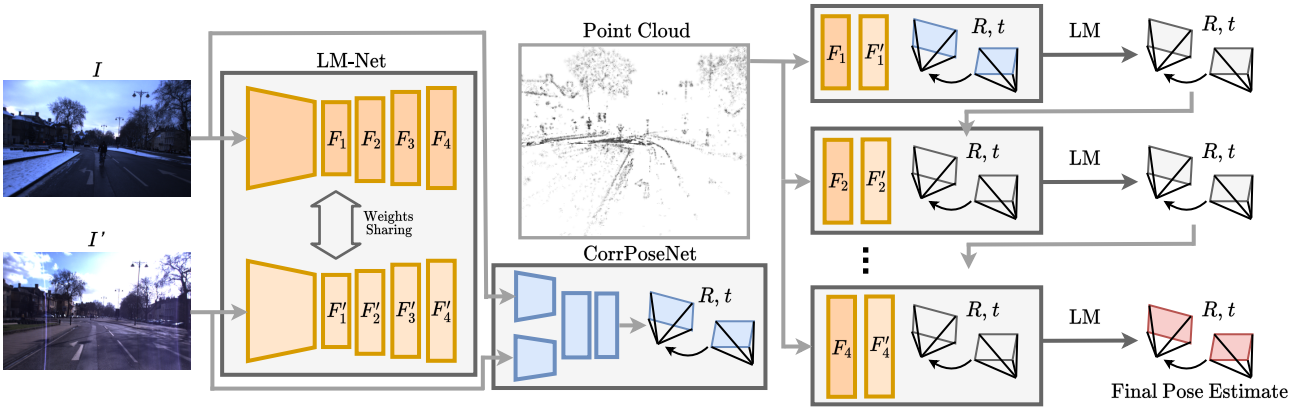


Figure 1: We propose LM-Reloc – a novel approach for visual relocalization based on direct image alignment. It consists of two deep neural networks: LM-Net, an encoder-decoder network for learning dense visual descriptors and a CorrPoseNet to bootstrap the direct image alignment. The final 6DoF relative pose estimate between image I and I' is obtained in a coarse-to-fine pyramid scheme leveraging the learned feature maps. The initialization for the direct image alignment is obtained by the CorrPoseNet.

Abstract

We present LM-Reloc – a novel approach for visual relocalization based on direct image alignment. In contrast to prior works that tackle the problem with a feature-based formulation, the proposed method does not rely on feature matching and RANSAC. Hence, the method can utilize not only corners but any region of the image with gradients. In particular, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. The learned features significantly improve the robustness of direct image alignment, especially for relocalization across different conditions. To further improve the robustness of LM-Net against large image baselines, we propose a pose estimation network, CorrPoseNet, which regresses the relative pose to bootstrap the direct image alignment. Evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark show that our approach delivers more accurate results than previous state-of-the-art methods while being comparable in terms of robustness.

1. Introduction

Map-based relocalization, that is, to localize a camera within a pre-built reference map, is becoming more and more important for robotics [6], autonomous driving [24, 4] and AR/VR [29]. Sequential-based approaches, which leverage the temporal structure of the scene provide more stable pose estimations and also deliver the positions in global coordinates compared to single image-based localization methods. The map is usually generated by either using LiDAR or visual Simultaneous Localization and Mapping (vSLAM) solutions. In this paper, we consider vSLAM maps due to the lower-cost visual sensors and the richer semantic information from the images. Feature-based methods [14, 7, 22, 23] and direct methods [13, 12, 11, 1] are two main lines of research for vSLAM.

Once a map is available, the problem of relocalizing within this map at any later point in time requires to deal with long-term changes in the environment. This makes a centimeter-accurate global localization challenging, especially in the presence of drastic lighting and appearance changes in the scene. For this task, feature-based methods are the most commonly used approaches to estimate the ego-pose and its orientation. This is mainly due to the advantage that features are more robust against changes in lighting/illumination in the scene.

However, feature-based methods can only utilize keypoints that have to be matched across the images before the pose estimation begins. Thus they ignore large parts of the

*Equal contribution.

available information. Direct methods, in contrast, can take advantage of all image regions with sufficient gradients and as a result, are known to be more accurate on visual odometry benchmarks [41, 11, 39].

In this paper, we propose LM-Reloc, which applies direct techniques to the task of relocalization. LM-Reloc consists of LM-Net, CorrPoseNet, and a non-linear optimizer, which work seamlessly together to deliver reliable pose estimation without RANSAC and feature matching. In particular, we derive a loss formulation, which is specifically designed to work well with the Levenberg-Marquardt (LM) algorithm [16, 20]. We use a deep neural network, LM-Net, to train descriptors that are being fed to the direct image alignment algorithm. Using these features results in better robustness against bad initializations, large baselines, and against illumination changes.

While the robustness improvements gained with our loss formulation are sufficient in many cases, for very large baselines or strong rotations, some initialization can still be necessary. To this end, we propose a pose estimation network. Based on two images it directly regresses the 6DoF pose, which we utilize as initialization for LM-Net. The CorrPoseNet contains a correlation layer as proposed in [27], which ensures that the network can handle large displacements. The proposed CorrPoseNet displays a lot of synergies with LM-Net. Despite being quite robust, the predictions of the CorrPoseNet are not very accurate. Thus it is best used in conjunction with our LM-Net, resulting in very robust and accurate pose estimates.

We evaluate our approach on the relocalization tracking benchmark from [36], which contains scenes simulated using CARLA [9], as well as sequences from the Oxford RobotCar dataset [19]. Our LM-Net shows superior accuracy especially in terms of rotation while being competitive in terms of robustness.

We summarize our main contributions:

- LM-Reloc, a novel pipeline for visual relocalization based on direct image alignment, which consists of LM-Net, CorrPoseNet, and a non-linear optimizer.
- A novel loss formulation together with a point sampling strategy that is used to train LM-Net such that the resulting feature descriptors are optimally suited to work with the LM algorithm.
- Extensive evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark which show that the proposed approach achieves state-of-the-art relocalization accuracy without relying on feature matching or RANSAC.

2. Related Work

In this section, we review the main topics that are closely related to our work, including direct methods for visual localization and feature-based visual localization methods.

Direct methods for visual localization. In recent years, direct methods [13, 12, 11] for SLAM and visual odometry have seen a great progress. Unlike feature-based methods [14, 7, 22, 23] which firstly extracts keypoints as well as the corresponding descriptors, and then minimize the geometric errors, direct methods minimize the energy function based on the photometric constancy assumption without performing feature matching or RANSAC. By utilizing more points from the images, direct methods show higher accuracy than feature-based methods [39]. However, classical direct methods show lower robustness than feature-based methods when the photometric constancy assumption is violated due to, e.g., the lighting and weather changes which are typical for long-term localization [33]. In [2] and [25], the authors propose to use the handcrafted features to improve the robustness of direct methods against low light or global appearance changes. Some recent works [5, 18, 36] address the issue by using learned features from deep neural networks [15]. In [5] they train deep features using a Hinge-Loss based on the Lucas-Kanade method, however, in contrast to us, they estimate the optical flow instead of applying the features to the task of relocalization. The most related work to ours is GN-Net [36] which proposes a Gauss-Newton loss to learn deep features. By performing direct image alignment on the learned features, GN-Net can deliver reliable pose estimation between the images taken from different weather or season conditions. The proposed LM-Net further derives the loss formulation based on Levenberg-Marquardt to improve the robustness against bad initialization compared to the Gauss-Newton method. Inspired by D3VO [38], LM-Reloc also proposes a relative pose estimation network with a correlation layer [27] to regress a pose estimate which is used as the initialization for the optimization.

Feature-based visual localization. Most approaches for relocalization utilize feature detectors and descriptors, which can either be handcrafted, such as SIFT [17] or ORB [28], or especially in the context of drastic lighting and appearance changes can be learned. Recently, many descriptor learning methods have been proposed which follow a *detect-and-describe* paradigm, e.g., SuperPoint [8], D2-Net [10], or R2D2 [26]. Moreover, SuperGlue [32], a learning-based alternative to the matching step of feature-based methods has been proposed and yields significant performance improvements. For a complete relocalization pipeline the local pose refinement part has to be preceded by finding the closest image in a database given a query [3]. While some approaches [31, 30, 35] address the joint problem, in this work, we decouple these two tasks and only focus on the pose refinement part.

3. Method

In this work, we address the problem of computing the 6DoF pose $\boldsymbol{\xi} \in SE(3)$ between two given images I and I'. Furthermore, we assume that depths for a sparse set of points P are available, e.g., by running a direct visual SLAM system such as DSO [11].

The overall pipeline of our approach is shown in Figure 1. It is composed of LM-Net, CorrPoseNet, and a non-linear optimizer using the LM algorithm. LM-Net is trained with a novel loss formulation designed to learn feature descriptors optimally suited for the LM algorithm. The encoder-decoder architecture takes as input a reference image I as well as a target image I'. The network is trained end-to-end and will produce multi-scale feature maps F_l and F'_l, where $l = 1, 2, 3, 4$ denotes the different levels of the feature pyramid. In order to obtain an initial pose estimate for the non-linear optimization, we propose CorrPoseNet, which takes I and I' as the inputs and regress their relative pose. Finally, the multi-scale feature maps together with the depths obtained from DSO [11] form the non-linear energy function which is minimized using LM algorithm in a coarse-to-fine manner to obtain the final relative pose estimate. In the following, we will describe the individual components of our approach in more detail.

3.1. Direct Image Alignment with Levenberg-Marquardt

In order to optimize the pose $\boldsymbol{\xi}$ (consisting of rotation matrix $\mathbf{R}$ and translation $\mathbf{t}$), we minimize the feature-metric error:

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{p} \in P} \left\| F'_l(\mathbf{p}') - F_l(\mathbf{p}) \right\|_\gamma , \tag{1}$$

where $||\cdot||_\gamma$ is the Huber norm and $\mathbf{p}'$ is the point projected onto the target image I' using the depths and the pose:

$$\mathbf{p}' = \Pi \left(\mathbf{R}\, \Pi^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t} \right). \tag{2}$$

This energy function is first minimized on the coarsest pyramid level 1, whose feature maps F_1 have a size of $(w/8, h/8)$, yielding a rough pose estimate. The estimate is refined by further minimizing the energy function on the subsequent pyramid levels 2, 3, and 4, where F_4 has the size of the original image (w, h). In the following, we provide details of the minimization performed in every level and for simplicity we will denote F_l as F from now on.

Minimization is performed using the Levenberg-Marquardt algorithm. In each iteration we compute the update $\boldsymbol{\delta} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$ as follows: Using the residual vector $\mathbf{r} \in \mathbb{R}^n$, the Huber weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, and the Jacobian of the residual vector with respect to the pose $\mathbf{J} \in \mathbb{R}^{n \times 6}$, we compute the Gauss-Newton system:

$$\mathbf{H} = \mathbf{J}^T \mathbf{W} \mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T \mathbf{W} \mathbf{r}. \tag{3}$$

The damped system can be obtained with either Levenberg's formula [16]:

$$\mathbf{H}' = \mathbf{H} + \lambda \mathbf{I} \tag{4}$$

or the Marquardt's formula [20]:

$$\mathbf{H}' = \mathbf{H} + \lambda \,\mathrm{diag}(\mathbf{H}) \tag{5}$$

depending on the specific application.

The update $\boldsymbol{\delta}$ and the pose $\boldsymbol{\xi}^i$ in the iteration i are computed as:

$$\boldsymbol{\delta} = \mathbf{H}'^{-1} \mathbf{b} \text{ and } \boldsymbol{\xi}^i = \boldsymbol{\delta} \boxplus \boldsymbol{\xi}^{i-1}, \tag{6}$$

where $\boxplus : \mathfrak{se}(3) \times SE(3) \to SE(3)$ is defined as in [11].

The parameter λ can be seen as an interpolation factor between gradient descent and the Gauss-Newton algorithm. When λ is high the method behaves like gradient descent with a small step size, and when it is low it is equivalent to the Gauss-Newton algorithm. In practice, we start with a relatively large λ and multiply it by 0.5 after a successful iteration, and by 4 after a failed iteration [11].

Figure 2 shows the typical behaviour of the algorithm. In the beginning the initial pose is inaccurate, resulting in projected point positions, which are a couple of pixels away from the correct location. λ will be high meaning that the algorithm will behave similar to gradient descent. After a couple of iterations, the pose got more accurate, and the projected points are in a closer vicinity to the correct location. By now, λ has probably decreased, so the algorithm will behave more similar to the Gauss-Newton algorithm. Now we expect the algorithm to converge quickly.

3.2. Loss Formulation for Levenberg-Marquardt

The key contribution of this work is LM-Net which provides feature maps F that improve the convergence behaviour of the LM algorithm and, in the meantime, are invariant to different conditions. We train our network in a Siamese fashion based on ground-truth pixel correspondences.

In this section, $\mathbf{p}$ denotes a reference point (located on image I) and the ground-truth correspondence (located on image I') is $\mathbf{p}'_{\text{gt}}$. For the loss functions explained below we further categorize $\mathbf{p}'$ into $\mathbf{p}'_{\text{neg}}$, $\mathbf{p}'_{\nabla}$, and $\mathbf{p}'_{\nabla^2}$, which is realized by using different negative correspondence sampling. Our loss formulation is inspired by the typical behaviour of the Levenberg-Marquardt algorithm explained in the previous section (see Figure 2). For a point, we distinguish four cases which can happen during the optimization:

1. The point is at the correct location ($\mathbf{p}'_{\text{gt}}$).

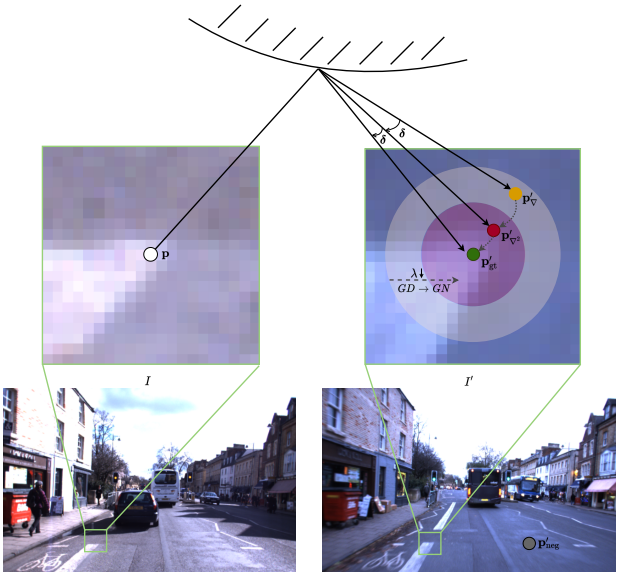


Figure 2: Visualization of the typical behavior of direct image alignment with Levenberg-Marquardt. Initially, the projected point position (orange point, $\mathbf{p}'_{\nabla}$) is far away from the correct solution (green point, $\mathbf{p}'_{\text{gt}}$), and λ is large, yielding an update step similar to gradient descent. After some iterations the projected point position gets closer to the optimum (red point, $\mathbf{p}'_{\nabla^2}$) and at the same time λ will get smaller, leading to an update step similar to the Gauss-Newton algorithm. This is the intuition behind our point sampling strategy, where we utilize the ground-truth correspondence $\mathbf{p}'_{\text{gt}}$ for Equation (7), a negative $\mathbf{p}'_{\text{neg}}$ sampled across the whole image for Equation (8), a negative $\mathbf{p}'_{\nabla}$ sampled in a far vicinity for Equation (12), and a negative $\mathbf{p}'_{\nabla^2}$ sampled in a close vicinity for Equation (14).

2. The point is an outlier ($\mathbf{p}'_{\text{neg}}$).
3. The point is relatively far from the correct solution ($\mathbf{p}'_{\nabla}$).
4. The point is very close to the correct solution ($\mathbf{p}'_{\nabla^2}$).

In the following we will derive a loss function for each of the 4 cases:

1. The point is already at the correct location. In this case we would like the residual to be as small as possible, in the best case 0.

$$E_{\text{pos}} = \|F'(\mathbf{p}'_{\text{gt}}) - F(\mathbf{p})\|^2 \tag{7}$$

2. The point is an outlier or the pose estimate is completely wrong. In this case the projected point position can be at a completely different location than the correct correspondence. In this scenario we would like the residual of this pixel to be very large to reflect this, and potentially reject a wrong update. To enforce this property we sample a negative correspondences $\mathbf{p}'_{\text{neg}}$ uniformly across the whole image, and compute

$$E_{\text{neg}} = \max\left(M - \|F'(\mathbf{p}'_{\text{neg}}) - F(\mathbf{p})\|^2, 0\right) \tag{8}$$

where M is the margin how large we would like the energy of a wrong correspondence to be. In practice, we set it to 1.

3. The predicted pose is relatively far away from the optimum, meaning that the projected point position will be a couple of pixels away from the correct location. As this typically happens during the beginning of the optimization we assume that λ will be relatively large and the algorithm behaves similar to gradient descent. In this case we want that the gradient of this point is oriented in the direction of the correct solution, so that the point has a positive influence on the update step.

For computing a loss function to enforce this property we sample a random negative correspondence $\mathbf{p}'_{\nabla}$ in a relatively large vicinity around the correct solution (in our experiments we use 5 pixels distance). Starting from this negative correspondence $\mathbf{p}'_{\nabla}$ we first compute the 2×2 Gauss-Newton system for this individual point, similarly to how it is done for optical flow estimation using Lucas-Kanade:

$$\mathbf{r}_{\mathbf{p}}(\mathbf{p}, \mathbf{p}'_{\nabla}) = \mathbf{F}'(\mathbf{p}'_{\nabla}) - \mathbf{F}(\mathbf{p}) \tag{9}$$

$$\mathbf{J}_{\mathbf{p}} = \frac{d\mathbf{F}'(\mathbf{p}'_{\nabla})}{d\mathbf{p}'_{\nabla}} \text{ and } \mathbf{H}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T\mathbf{J}_{\mathbf{p}} \text{ and } \mathbf{b}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T\mathbf{r}_{\mathbf{p}} \tag{10}$$

We compute the damped system using a relatively large fixed λ_f, as well as the optical flow step*

$$\mathbf{H}'_{\mathbf{p}} = \mathbf{H}_{\mathbf{p}} + \lambda_f\,\mathbf{I} \text{ and } \mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla} + \mathbf{H}'^{-1}_{\mathbf{p}}\mathbf{b}_{\mathbf{p}}. \tag{11}$$

In order for this point to have a useful contribution to the direct image alignment, this update step should move in the correct direction by at least δ. We enforce this using a Gradient-Descent loss function which is small only if the distance to the correct correspondence *after* the update is smaller than before the update:

$$E_{\text{GD}} = \max\left(\|\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\|^2 - \|\mathbf{p}'_{\nabla} - \mathbf{p}'_{\text{gt}}\|^2 + \delta, 0\right) \tag{12}$$

In practice, we choose $\lambda_f = 2.0$ and $\delta = 0.1$.

4. The predicted pose is very close to the optimum, yielding a projected point position in very close proximity of the correct correspondence, and typically λ will be very small, so the update will mostly be a Gauss-Newton step. In this case we would like the algorithm to converge as quickly as possible, with subpixel accuracy. We enforce this using the Gauss-Newton loss [36]. To compute it we first sample a random negative correspondence $\mathbf{p}'_{\nabla^2}$ in a 1-pixel vicinity

*Here we use Equation (4) instead of Equation (5) since we find it more stable for training LM-Net.

around the correct location. Then we use Equations (9) and (10), replacing $\mathbf{p}'_{\nabla}$ with $\mathbf{p}'_{\nabla^2}$ to obtain the Gauss-Newton system formed by $\mathbf{H}_\mathbf{p}$ and $\mathbf{b}_\mathbf{p}$. We compute the updated pixel location:

$$\mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla^2} + (\mathbf{H}_\mathbf{p} + \epsilon\,\mathbf{I})^{-1}\mathbf{b}_\mathbf{p} \tag{13}$$

Note that in contrast to the computation of the LM-Loss (Equation (12)), in this case ϵ is just added to ensure invertibility and therefore ϵ is much smaller than the λ_f used above. The Gauss-Newton loss is computed with:

$$\begin{aligned} E_{\text{GN}} = \frac{1}{2}(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}})^T \mathbf{H}_\mathbf{p} (\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}) \\ + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}_\mathbf{p}|) \end{aligned} \tag{14}$$

Note how all our 4 loss components use a different way to sample the involved points, depicted also in Figure 2. With the derivation above we argue that each loss component is important to achieve optimal performance and we demonstrate this in the results section. Note that the Gauss-Newton systems computed for the GD-Loss and the GN-Loss are very relevant for the application of direct image alignment. In fact the full Gauss-Newton system containing all points (Equation (3)), can be computed from these individual Gauss-Newton systems (Equation (10)) by simply summing them up and multiplying them with the derivative with respect to the pose [36].

3.3. CorrPoseNet

In order to deal with the large baselines between the images, we propose CorrPoseNet to regress the relative pose between two images I and I', which serves as the initialization of LM optimization. As our network shall work even in cases of large baselines and strong rotations, we utilize the correlation layer proposed in [27] which is known to boost the performance of affine image transformation and optical flow [21] estimation for large displacements, but has not been applied to pose estimation before.

Our network first computes deep features $\mathbf{f}_{\text{corr}}, \mathbf{f}'_{\text{corr}} \in \mathbb{R}^{h\times w\times c}$ from both images individually using multiple strided convolutions with ReLU activations in between. Then the correlation layer correlates each pixel from the normalized source features with each pixel from the normalized target features yielding the correlation map $\mathbf{c} \in \mathbb{R}^{h\times w\times(h\times w)}$:

$$\mathbf{c}(i,j,(i',j')) = \mathbf{f}_{\text{corr}}(i,j)^T \mathbf{f}'_{\text{corr}}(i',j') \tag{15}$$

The correlation map is then normalized in the channel dimension and fed into 2 convolutional layers each followed by batch norm and ReLU. Finally we regress the Euler angle $\mathbf{r}^{\text{euler}}$ and translation $\mathbf{t}$ using a fully connected layer. More details on the architecture are shown in the supplementary material.

We train CorrPoseNet from scratch with image pairs and groundtruth poses $\mathbf{r}^{\text{euler}}_{\text{gt}}, \mathbf{t}_{\text{gt}}$. We utilize an L2-loss working directly on Euler angles and translation:

$$E = \|\mathbf{t} - \mathbf{t}_{\text{gt}}\|_2 + \lambda\|\mathbf{r}^{\text{euler}} - \mathbf{r}^{\text{euler}}_{\text{gt}}\|_2, \tag{16}$$

where λ is the weight, which we set to 10 in practice.

As the distribution of groundtruth poses in the Oxford training data is limited we apply the following data augmentation. We first generate dense depths for all training images using a state-of-the-art dense stereo matching algorithm [40]. The resulting depths are then used to warp the images to a different pose sampled from a uniform distribution. In detail, we first warp the depth image to the random target pose, then inpaint the depth image using the OpenCV implementation of Navier Stokes, and finally warp our image to the target pose using this depth map. Note that the dense depths are only necessary for training, not for evaluation. We show an ablation study on the usage of correlation layers and the proposed data augmentation in the supplementary material.

4. Experiments

We evaluate our method on the relocalization tracking benchmark proposed in [36], which contains images created with the CARLA simulator [9], and scenes from the Oxford RobotCar dataset [19]. We train our method on the respective datasets from scratch. LM-Net is trained using the Adam optimizer with a learning rate of 10^{-6} and for CorrPoseNet we use a learning rate of 10^{-4}. For both networks we choose hyperparameters and epoch based on the results on the validation data. Our networks use the same hyperparameters for all experiments except where stated otherwise; the direct image alignment code is slightly adapted for Oxford RobotCar, mainly to improve performance when the ego-vehicle is standing still.

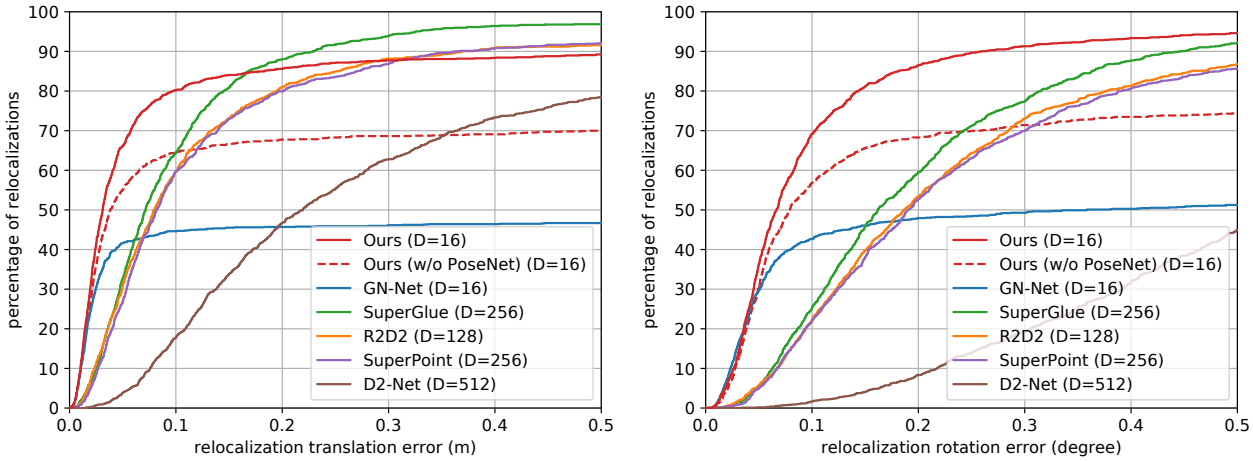
As the original relocalization tracking benchmark [36] does not include validation data on Oxford RobotCar we have manually aligned two new sequences, namely *2015-04-17-09-06-25* and *2015-05-19-14-06-38*, and extend the benchmark with these sequences as validation data.

Evaluation metrics: We evaluate the predicted translation $\mathbf{t}_{\text{est}}$ and rotation $\mathbf{R}_{\text{est}}$ against the ground-truth $\mathbf{t}_{\text{gt}}$ and $\mathbf{R}_{\text{gt}}$ according to Equations (17) and (18).

$$t_\Delta = \|\mathbf{t}_{\text{est}} - \mathbf{t}_{\text{gt}}\|_2 \tag{17}$$

$$R_\Delta = \arccos\left(\frac{\text{trace}(\mathbf{R}^{-1}_{\text{est}}\mathbf{R}_{\text{gt}}) - 1}{2}\right) \tag{18}$$

In this section, we plot the cumulative translation and rotation error until 0.5m and $0.5°$, respectively. For quantitative results we compute the area under curve (AUC) of these



(a) Translation error.

(b) Rotation error.

Figure 3: Results on the CARLA relocalization tracking benchmark test data [36]. For each error threshold we show the percentage of relocalizations (cumulative error plot) for LM-Reloc (ours) and other state-of-the-art methods. Compared to the indirect methods our approach exhibits significantly better accuracy in both translation and rotation, while having a similar robustness. Compared to GN-Net, the novel loss formulation (see red dashed line), and the CorrPoseNet (see red line) both boost the robustness. D is the feature dimensionality.

cumulative curves in percent, which we denote as t_{AUC} for translation and R_{AUC} for rotation from now on.

We evaluate the following direct methods:

Ours: The full LM-Reloc approach consisting of CorrPoseNet, LM-Net features and direct image alignment based on Levenberg-Marquardt. The depths used for the image alignment are estimated with the stereo version [37] of DSO [11].

Ours (w/o CorrPoseNet): For a more fair comparison to GN-Net we use identity as initialization for the direct image alignment instead of CorrPoseNet. This enables a direct comparison between the two loss formulations.

GN-Net [36]: In this work, we have also improved the parameters of the direct image alignment pipeline based on DSO [11]. Thus we have re-evaluated GN-Net with this improved pipeline to make the comparison as fair as possible. These re-evaluated results are better than the results computed in the original GN-Net paper.

Baseline methods: Additionally, we evaluate against current state-of-the-art indirect methods, namely SuperGlue [32], R2D2 [26], SuperPoint [8], and D2-Net [10]. For these methods, we estimate the relative pose using the models provided by the authors and the OpenCV implementation of solvePnPRansac. We have tuned the parameters of RANSAC on the validation data and used 1000 iterations and a reprojection error threshold of 3 for all methods. For estimating depth values at keypoint locations we use OpenCV stereo matching. It would be possible to achieve a higher accuracy by using SfM and MVS solutions such as COLMAP [34]. However, one important disadvantage of these approaches is, that building a map is rather time consuming and computationally expensive, whereas all other approaches evaluated on the benchmark [36] are able to create the map close to real-time, enabling applications like long-term loop-closure and map-merging.

Table 1: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the CARLA relocalization tracking benchmark test data. Powered by our novel loss formulation and the combination with CorrPoseNet, LM-Reloc achieves lower rotation and translation errors compared to the state-of-the-art.

Method	t_{AUC}	R_{AUC}
Ours	**80.65**	**77.83**
SuperGlue [32]	78.99	59.31
R2D2 [26]	73.47	54.42
SuperPoint [8]	72.76	53.38
D2-Net [10]	47.62	16.47
Ours (w/o CorrPoseNet)	63.88	61.9
GN-Net [36]	43.72	44.08

4.1. CARLA Relocalization Benchmark

Figure 3 depicts the results on the test data of the CARLA benchmark. For all methods we show the cumulative error plot for translation in meters and rotation in degree. It can be seen that our method is more accurate than the state-of-the-art while performing similarly in terms of robustness. We also show the AUC for the two Figures in Table 1. Compared to GN-Net it can be seen that our new loss formulation significantly improves the results,

Table 2: Results on the Oxford RobotCar relocalization tracking benchmark [36]. We compare LM-Net (Ours) against other state-of-the-art methods (SuperGlue, R2D2, SuperPoint, and D2-Net). As can be seen from the results, our method almost consistently outperforms other SOTA approaches in terms of rotation AUC whilst achieving comparable results on translation AUC.

Sequence	Ours		SuperGlue [32]		R2D2 [26]		SuperPoint [8]		D2-Net [10]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	79.83	**55.48**	**81.01**	52.83	80.86	53.57	78.95	50.03	71.93	39.0
Sunny-Rainy	71.54	**43.7**	**75.58**	40.59	74.84	41.23	69.76	37.12	65.63	27.5
Sunny-Snowy	59.69	**44.06**	**63.57**	43.64	62.92	41.78	60.85	40.02	55.65	30.86
Overcast-Rainy	80.54	**63.7**	79.99	61.64	**81.29**	61.23	80.36	61.56	75.66	51.06
Overcast-Snowy	55.38	47.88	57.67	47.16	**57.68**	**48.41**	55.39	44.96	51.17	34.54
Rainy-Snowy	68.57	**41.67**	69.91	39.87	**71.79**	39.86	67.7	38.05	61.91	27.74

Table 3: This table shows the results on the Oxford RobotCar relocalization tracking benchmark test data against GN-Net. Thanks to our LM-based loss formulation we consistently outperform GN-Net on all sequences.

Sequence	Ours (w/o CorrPoseNet)		GN-Net [36]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	**79.61**	**55.45**	73.53	49.31
Sunny-Rainy	**70.46**	**42.86**	64.58	37.27
Sunny-Snowy	**59.7**	**44.17**	55.27	41.36
Overcast-Rainy	**79.67**	**63.08**	75.72	60.13
Overcast-Snowy	**54.94**	**47.19**	51.34	42.91
Rainy-Snowy	**66.23**	**39.93**	62.63	36.2

even when used without the CorrPoseNet as initialization. The figure conveys that the direct methods (Ours, GN-Net) are more accurate than the evaluated indirect methods.

4.2. Oxford RobotCar Relocalization Benchmark

We compare to the state-of-the-art indirect methods on the 6 test sequence pairs consisting of the sequences *2015-02-24-12-32-19* (sunny), *2015-03-17-11-08-44* (overcast), *2014-12-05-11-09-10* (rainy), and *2015-02-03-08-45-10* (snowy). In Table 2, we show the area under curve until 0.5 meters / 0.5 degrees for all methods. It can be seen that our method clearly outperforms the state-of-the-art in terms of rotation accuracy, while being competitive in terms of translation error. It should be noted that the ground-truth for these sequences was generated using ICP alignment of the 2D-LiDAR data accumulated for 60 meters. We have computed that the average root mean square error of the ICP alignment is 16 centimeters. Therefore, especially the ground-truth translations have limited accuracy. As can be seen from Figure 3, the accuracy improvements our method provides are especially visible in the range below 0.15 meters which is hard to measure on this dataset. The rotation error of LiDAR alignment is lower than the translational

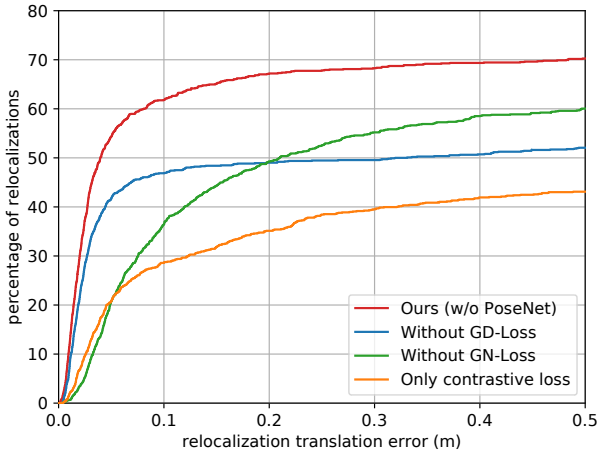


Figure 4: This plot shows our ablation study for removing different loss parts on the CARLA relocalization tracking benchmark. Without the GD-loss the achieved robustness is reduced, whereas removing the GN-loss leads to decreased accuracy. Using our full loss formulation yields a large improvement.

one, which is why we clearly observe the improvements of our method on the rotations.

In Table 3, we compare LM-Net without the CorrPoseNet to GN-Net. Due to our novel loss formulation LM-Net outperforms the competitor on all sequences significantly.

4.3. Ablation Studies

We evaluate LM-Net on the CARLA validation data with and without the various losses (Figure 4). Compared to a normal contrastive loss, the given loss formulation is a large improvement. As expected, E_{GD} (green line) mainly improves the robustness, whereas E_{GN} (blue line) improves the accuracy. Only when used together (our method) we achieve large robustness and large accuracy, confirming our theoretical derivation in Section 3.

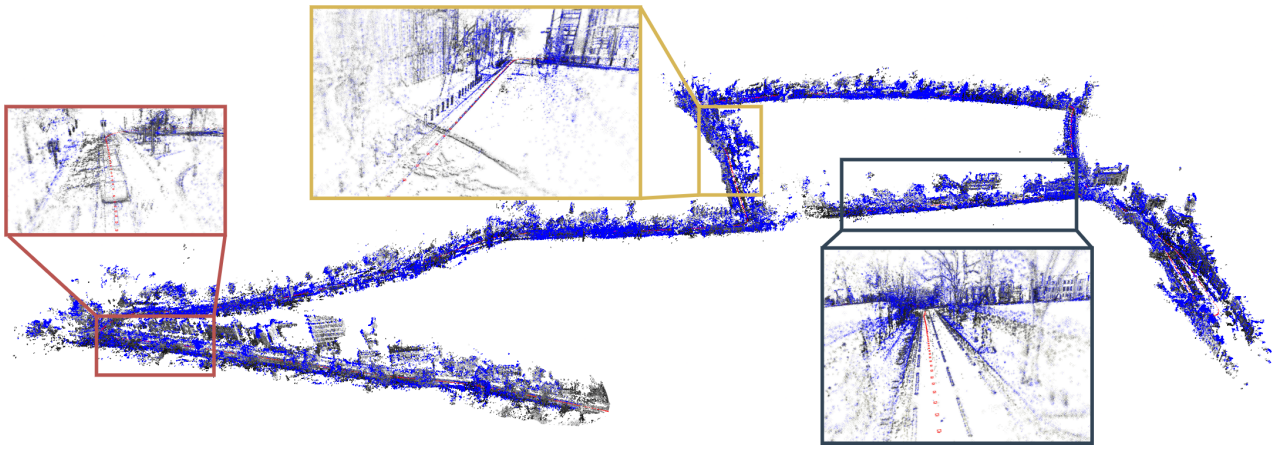


Figure 5: This figure shows a point cloud from a sunny reference map (grey points) overlayed with the point cloud from a relocalized snowy sequence (blue points). The well aligned point clouds demonstrate the high relocalization accuracy of LM-Reloc.

Figure 6: Example image pairs from the relocalization tracking benchmark which have been successfully relocalized by LM-Reloc (with an accuracy of better than 10 cm). Top row: Oxford sunny against snowy condition, middle row: Oxford sunny against rainy condition, bottom row: CARLA benchmark.

4.4. Qualitative Results

To demonstrate the accuracy of our approach in practice, we show qualitative results on the Oxford RobotCar dataset. We track the snowy test sequence *2015-02-03-08-45-10* using Stereo DSO [37] and at the same time perform relocalization against the sunny reference map *2015-02-24-12-32-19*. Relocalization between the current keyframe and the closest map image is performed using LM-Net. Initially, we give the algorithm the first corresponding map image (which would in practice be provided by an image retrieval approach such as NetVLAD [3]). Afterwards we find the closest map image for each keyframe using the previous solution for the transformation between the map and the current SLAM world T_{w_m}. We visualize the current point cloud (blue) and the point cloud from the map (grey) overlayed using the smoothed T_{w_m} (Figure 5). The point clouds will align only if the relocalization is accurate. As can be seen in Figure 5, the lane markings, poles, and buildings between the reference and query map align well, hence qualitatively showing the high relocalization accuracy of our method. We recommend watching the video at `https://vision.in.tum.de/lm-reloc`. In Figure 6 we show example images from the benchmark.

5. Conclusion

We have presented LM-Reloc as a novel approach for direct visual localization. In order to estimate the relative 6DoF pose between two images from different conditions, our approach performs direct image alignment on the trained features from LM-Net without relying on feature matching or RANSAC. In particular, with the loss function designed seamlessly for the Levenberg-Marquart algorithm, LM-Net provides deep feature maps that coin the characteristics of direct image alignment and are also invariant to changes in lighting and appearance of the scene. The experiments on the CARLA and Oxford RobotCar relocalization tracking benchmark exhibit the state-of-the-art performance of our approach. In addition, the ablation studies also show the effectiveness of the different components of LM-Reloc.

References

[1] H. Alismail, B. Browning, and S. Lucey. Photometric bundle adjustment for vision-based SLAM. In *ACCV*, 2017.

[2] H. Alismail, M. Kaess, B. Browning, and S. Lucey. Direct visual odometry in low light using binary descriptors. *RA-L*, 2, 2017.

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016.

[4] M. A. Brubaker, A. Geiger, and R. Urtasun. Map-based probabilistic visual self-localization. *PAMI*, 38(4):652–665, 2015.

[5] C.-H. Chang, C.-N. Chou, and E. Y. Chang. CLKN: Cascaded lucas-kanade networks for image alignment. In *CVPR*, pages 2213–2221, 2017.

[6] M. Cummins and P. Newman. FAB-MAP: probabilistic localization and mapping in the space of appearance. *IJRR*, 27(6), 2008.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *PAMI*, 29(6):1052–1067, 2007.

[8] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: self-supervised interest point detection and description. In *CVPRW*, 2018.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: an open urban driving simulator. In *CoRL*, 2017.

[10] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: a trainable CNN for joint description and detection of local features. In *CVPR*, 2019.

[11] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *PAMI*, 40(3), 2018.

[12] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014.

[13] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IROS*, pages 2100–2106. IEEE, 2013.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234. IEEE, 2007.

[15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

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

[18] Z. Lv, F. Dellaert, J. M. Rehg, and A. Geiger. Taking a deeper look at the inverse compositional algorithm. In *CVPR*, pages 4581–4590, 2019.

[19] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1), 2017.

[20] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[21] I. Melekhov, A. Tiulpin, T. Sattler, M. Pollefeys, E. Rahtu, and J. Kannala. Dgc-net: Dense geometric correspondence network. In *WACV*, pages 1034–1042. IEEE, 2019.

[22] R. Mur-Artal, J. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE T-RO*, 31(5), 2015.

[23] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE T-RO*, 33(5), 2017.

[24] T. Ort, L. Paull, and D. Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. In *ICRA*, pages 2040–2047. IEEE, 2018.

[25] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman. Nid-slam: Robust monocular slam using normalised information distance. In *CVPR*, pages 1435–1444, 2017.

[26] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. In *NeurIPS*, pages 12405–12415, 2019.

[27] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, pages 6148–6157, 2017.

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

[29] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, et al. Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality. *Proceedings of the IEEE*, 106(11):2020–2039, 2018.

[30] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019.

[31] P.-E. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *CoRL*, 2018.

[32] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020.

[33] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018.

[34] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[35] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, pages 7199–7209, 2018.

[36] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. *RA-L*, 5(2):890–897, 2020.

[37] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *ICCV*, 2017.

[38] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers. D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, pages 1281–1292, 2020.

[39] N. Yang, R. Wang, X. Gao, and D. Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *RA-L*, 3(4):2878–2885, 2018.

[40] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pages 185–194, 2019.

[41] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis. Photometric patch-based visual-inertial odometry. In *ICRA*, pages 3264–3271, 2017.

List of Figures

1.1 DM-VIO (Chapter 4) running live on a laptop, connected to a monocular camera with integrated IMU. It reconstructs the trajectory and 3D environment (pointcloud on the laptop) with remarkable accuracy in real-time. The source-code for this live demo has been released, see Appendix B. 15
1.2 Example of how a quadrocopter can autonomously explore an unknown environment, utilizing a visual SLAM system. This work was published in [1] and is based on the Bachelor's thesis [57] of Lukas von Stumberg. 20
1.3 Accurate poses from SLAM can also benefit downstream computer vision tasks. MonoRec [3], a work led by Felix Wimbauer, achieves accurate 3D reconstruction using a cost volume, relying on accurate poses from visual odometry. 21

2.1 Trajectories and pointclouds estimated by DM-VIO. Top: Magistrale of our office building reconstructed using a sequence from the TUM-VI dataset. Bottom: A sequence from the automotive 4Seasons dataset. Long stretches of constant velocity constitute a challenge for monocular visual-inertial odometry, but thanks to our novel IMU initializer, the method works very well. 27
2.2 Top: Example images from the relocalization tracking benchmark. Bottom: Deep features created by our network LM-Net from these images. Using these features, direct image alignment works well despite strong illumination and weather changes. 29
2.3 Relocalization demo on the Oxford RobotCar dataset. We relocalize sequences with different weather conditions (sunny and snowy in this example) and overlay the pointclouds using the estimated transformations. The pointclouds align very well, showing that the relocalization is accurate. 30

3.1 Simple factor graph which visualizes the probability density shown in Equation 3.16, and also the corresponding energy function given in Equation 3.17. There is a circle for each variable. Each summand of the energy function is represented by a square, connected to the variables, it depends on. 35
3.2 Factor graphs before and after marginalization of variable x. The marginalization factor replaces all connected factors and is connected to the variables which form the Markov blanket of x. 37

3.3 *Left:* Factor graph for the bundle adjustment performed in DSO. $\mathbf{P}_i$ is the pose of keyframe i, and d_i is the inverse depth of point i. In this example, there are three keyframes with one point each (d_i is hosted in $\mathbf{KF}_i$ respectively). Each residual depends on the host keyframe, the inverse depth of the point and the target keyframe which the point is projected into. *Right:* System after the Schur-complement trick used to speed up computation of the Gauss-Newton step. It works by "marginalizing" all depth values, which is very efficient, because the depths values are independent. After solving the system on the right for poses, the update for depth values is resubstituted. 39

3.4 Marginalization of $\mathbf{KF}_0$. This involves multiple steps in order to retain the sparsity of the system. 40

3.5 Interactions between the different optimizations present in DSO. While the *bundle adjustment* achieves high accuracy, it requires a good initialization and is only performed for keyframes. The *coarse tracking* estimates the pose of each frame using 2-frame direct image alignment against the latest keyframe utilizing the accurate depths from the bundle adjustment. Using the pose estimate from the coarse tracking, *point tracing* performs epipolar line search for a set of candidate points to obtain a rough depth value. When a new keyframe shall be added to the bundle adjustment, its pose is initialized with the result from coarse tracking, and depth values for new points are initialized with the result from point tracing. 41

3.6 Factor graph for the coarse tracking (2-frame direct image alignment). 42

4.1 In this paper, we propose a novel method for monocular visual-inertial odometry. It provides state-of-the-art performance on three different benchmarks. Here we show pointclouds and trajectories (red) for magistrale5, V203_difficult, and neighbor_2020-03-26_13-32-55_0. 47

4.2 Delayed Marginalization and PGBA: a) Normal marginalization in the visual graph. Note that not always the oldest pose is marginalized. b) Delayed marginalization: We marginalize all variables in the same order as the main graph, but with a delay d (in practice $d = 100$). Marginalization in this graph is equally fast as marginalizing in the main graph. c) For the pose graph bundle adjustment (PGBA) we populate the delayed graph with IMU factors. This optimization leverages the full photometric uncertainty. d) We readvance the marginalization in the graph used for PGBA to obtain an updated marginalization prior for the main system. This transfers inertial information from the initializer to the main system. 53

4.3 Our multi-stage IMU initialization. First we perform a coarse IMU initialization, which provides initial values for the PGBA. The PGBA captures the full visual covariances, achieving very accurate initial estimates for scale, gravity direction, and biases. It also provides an updated marginalization prior for the main graph. By also optimizing the scale in the main VIO system (green box), we can initialize early (purple box) and later reinitialize or perform marginalization replacement, if new information about the scale becomes available. The proposed delayed marginalization is what enables both, the PGBA, and the marginalization replacement. 56

4.4 Cumulative error plot for the TUM-VI dataset (drift in %). Our method clearly outperforms both VI-DSO and ORB-SLAM3 in terms of robustness. Thanks to its powerful loop closure system, ORB-SLAM3 has an advantage in terms of accuracy on some sequences. . 61

4.5 Cumulative error plot for the 4Seasons dataset (drift in %). With lots of stretches with constant velocity, this dataset is extremely challenging for monocular visual-inertial methods. Thanks to our novel IMU initializer powered by delayed marginalization and PGBA, DM-VIO is able to cope with it and even outperforms stereo-inertial methods. 62

5.1 We relocalize a snowy sequence from the Oxford RobotCar dataset in a pre-built map created using a sunny weather condition. The points from the prior map (gray) well align with the new points from the current run (blue), indicating that the relocalization is indeed accurate. 67

5.2 This figure shows training correspondences between a pair of images from our benchmark. 71

5.3 This figure shows images and their corresponding feature maps predicted by our GN-Net for the Oxford RobotCar dataset. Each column depicts the image and feature map for a sample taken from 2 different sequences. Despite lighting and weather changes, the feature maps are robust to these variations. The visualization of the features shows the high-dimensional descriptors reduced to 3D through PCA. 75

5.4 This figure shows the cumulative relocalization accuracy on the Oxford RobotCar dataset for different sequences. D denotes the dimension of the feature descriptor. Our method achieves the highest accuracy across all sequences. It is interesting to observe that despite being trained only on two sequences in overcast and sunny condition, our model still generalizes very well to even *unseen* rainy and snowy conditions. Even the model trained only on the synthetic CARLA benchmark outperforms all baselines, showing exceptional generalization capabilities. 79

5.5 This figure shows a point cloud result of our GN-Net. We relocalize a rainy sequence (blue) against a reference map created from the sunny sequence (gray). 80

5.6 Top: relocalization using the model trained with only the contrastive loss. Bottom: relocalization using the model trained with our loss formulation. This visually demonstrates the influence of the Gauss-Newton loss. 81

5.7 shows image pairs used in the qualitative relocalizations. Left: rainy (top row) and snowy (bottom row) images relocalized against the sunny reference images (right). 82

6.1 We propose LM-Reloc – a novel approach for visual relocalization based on direct image alignment. It consists of two deep neural networks: LM-Net, an encoder-decoder network for learning dense visual descriptors and a CorrPoseNet to bootstrap the direct image alignment. The final 6DoF relative pose estimate between image I and I' is obtained in a coarse-to-fine pyramid scheme leveraging the learned feature maps. The initialization for the direct image alignment is obtained by the CorrPoseNet. 87

6.2 Visualization of the typical behavior of direct image alignment with Levenberg-Marquardt. Initially, the projected point position (orange point, $\mathbf{p}'_{\nabla}$) is far away from the correct solution (green point, $\mathbf{p}'_{\text{gt}}$), and λ is large, yielding an update step similar to gradient descent. After some iterations the projected point position gets closer to the optimum (red point, $\mathbf{p}'_{\nabla^2}$) and at the same time λ will get smaller, leading to an update step similar to the Gauss-Newton algorithm. This is the intuition behind our point sampling strategy, where we utilize the ground-truth correspondence $\mathbf{p}'_{\text{gt}}$ for Equation (6.7), a negative $\mathbf{p}'_{\text{neg}}$ sampled across the whole image for Equation (6.8), a negative $\mathbf{p}'_{\nabla}$ sampled in a far vicinity for Equation (6.12), and a negative $\mathbf{p}'_{\nabla^2}$ sampled in a close vicinity for Equation (6.14). 92

6.3 Results on the CARLA relocalization tracking benchmark test data [5]. For each error threshold we show the percentage of relocalizations (cumulative error plot) for LM-Reloc (ours) and other state-of-the-art methods. Compared to the indirect methods our approach exhibits significantly better accuracy in both translation and rotation, while having a similar robustness. Compared to GN-Net, the novel loss formulation (see red dashed line), and the CorrPoseNet (see red line) both boost the robustness. D is the feature dimensionality. 97

6.4 This plot shows our ablation study for removing different loss parts on the CARLA relocalization tracking benchmark. Without the GD-loss the achieved robustness is reduced, whereas removing the GN-loss leads to decreased accuracy. Using our full loss formulation yields a large improvement. 101

6.5 This figure shows a point cloud from a sunny reference map (grey points) overlayed with the point cloud from a relocalized snowy sequence (blue points). The well aligned point clouds demonstrate the high relocalization accuracy of LM-Reloc. 102

6.6 Example image pairs from the relocalization tracking benchmark which have been successfully relocalized by LM-Reloc (with an accuracy of better than 10 cm). Top row: Oxford sunny against snowy condition, middle row: Oxford sunny against rainy condition, bottom row: CARLA benchmark. 103

7.1 Bottom: Example images from the EuRoC-dataset: Low illumination, strong motion blur and little texture impose significant challenges for odometry estimation. Still our method is able to process all sequences with a rmse of less then 0.23m. Top: Reconstruction, estimated pose (red camera) and groundtruth pose (green camera) at the end of V1_03_difficult. 109

7.2 Factor graphs for the visual-inertial joint optimization before and after the marginalization of a keyframe. 114

7.3 Partitioning of the factor graph from Fig. 7.2a into G_{metric} and G_{visual}. G_{metric} contains all IMU-factors while G_{visual} contains the factors that do not depend on $\boldsymbol{\xi}_{m\_d}$. Note that both of them do not contain any marginalization factors. 116

7.4 The scale estimation running on the V1_03_difficult sequence from the EuRoC dataset. We show the current scale estimate (bold blue), the groundtruth scale (bold red) and the current scale interval (light lines). The vertical dotted lines denote when the side changes (blue) and when the boundary of the scale interval is exceeded (red). In practice this means that M_{curr} contains the inertial factors since the last blue or red dotted line that is before the last red dotted line. For example at 16s it contains all inertial data since the blue line at 9 seconds. 117

7.5 rmse for different methods run 10 times (lines) on each sequence (columns) of the EuRoC dataset. 120

7.6 Cumulative error plot on the EuRoC-dataset (RT means realtime). This experiment demonstrates that the additional IMU not only provides a reliable scale estimate, but that it also significantly increases accuracy and robustness. 121

7.7 Relative Pose Error evaluated on three sequences of the EuRoC-dataset for visual-inertial ORB-SLAM [37], visual-inertial stereo LSD-SLAM [41] and our method. Although the proposed VI-DSO does not use loop closuring (like [37]) or stereo (like [41]), VI-DSO is quite competitive in terms of accuracy and robustness. Note that [37] with loop closures is slightly more accurate on average, yet it entirely failed on V1_03_difficult. 124

7.8 Scale estimate for MH_04_difficult (median result of 10 runs in terms of tracking accuracy). Note how the estimated scale converges to the correct value despite being initialized far from the optimum. 125

8.1 We propose D3VO – a novel monocular visual odometry (VO) framework which exploits deep neural networks on three levels: **D**eep depth (D), **D**eep pose (T_t^{t-1}) and **D**eep uncertainty (Σ) estimation. D3VO integrates the three estimations tightly into both the front-end tracking and the back-end non-linear optimization of a sparse direct odometry framework [15]. 129

8.2 Examples of point clouds and trajectories delivered by D3VO on KITTI Odometry Seq. 00, EuRoC *MH_05_difficult* and *V1_03_difficult*. The insets on EuRoC show the scenarios with low illumination and motion blur which are among the main reasons causing failures of traditional purely vision-based VO systems. 133

8.3 Examples of affine brightness transformation on EuRoC MAV [70]. Originally the source image ($I_{t'}$) and the target image (I_t) show different brightness. With the predicted parameters a, b, the transformed target images $I^{a',b'}$ have similar brightness as the source images, which facilitates the self-supervised training based on the brightness constancy assumption. 135
8.4 Qualitative results from KITTI and EuRoC MAV. The original image, the predicted depth maps and the uncertainty maps are shown from the left to the right, respectively. In particular, the network is able to predict high uncertainty on object boundaries, moving objects, highly reflecting and high frequency areas. 140
8.5 Qualitative comparison of the trajectories on *MH_05_difficult* and *V1_03_difficult* from EuRoC MAV. 144

List of Tables

2.1 Chronological list of publications related to this thesis. These include 11 full-length papers published in highly-ranked conferences and journals, as well as 2 book chapters. The works which are not part of this dissertation are shown in gray. 24

4.1 Evaluation of various mono (M) and stereo (S) visual-inertial odometry systems on EuRoC. Our system provides a notable improvement over the state-of-the art. Please note that a full SLAM system utilizing loop closures can achieve even more accurate results, e.g. ORB-SLAM-VI has a mean error of 0.075, and ORB-SLAM3 has a mean error of 0.043. 59

4.2 RMSE ATE in m on the TUM-VI dataset [20]. Best results in bold, underline is the best result among monocular methods. DM-VIO outperforms even state-of-the-art stereo-inertial methods by a large margin. 60

6.1 This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the CARLA relocalization tracking benchmark test data. Powered by our novel loss formulation and the combination with CorrPoseNet, LM-Reloc achieves lower rotation and translation errors compared to the state-of-the-art. 98

6.2 Results on the Oxford RobotCar relocalization tracking benchmark [5]. We compare LM-Net (Ours) against other state-of-the-art methods (SuperGlue, R2D2, SuperPoint, and D2-Net). As can be seen from the results, our method almost consistently outperforms other SOTA approaches in terms of rotation AUC whilst achieving comparable results on translation AUC. 99

6.3 This table shows the results on the Oxford RobotCar relocalization tracking benchmark test data against GN-Net. Thanks to our LM-based loss formulation we consistently outperform GN-Net on all sequences. 101

7.1 Accuracy of the estimated trajectory on the EuRoC dataset for several methods. Note that ORB-SLAM does a convincing job showing leading performance on some of the sequences. Nevertheless, since our method directly works on the sensor data (colors and IMU measurements), we observe similar precision and a better robustness – even without loop closuring. Moreover, the proposed method is the only one not to fail on any of the sequences. 123

8.1 Depth evaluation results on the KITTI Eigen split [150]. M: self-supervised monocular supervision; S: self-supervised stereo supervision; D: ground-truth depth supervison; D*: sparse auxiliary depth supervision. The upper part shows the comparison with the SOTA self-supervised network Monodepth2 [141] under the same setting and the ablation study of the brightness transformation parameters (*ab*) and the photometric uncertainty (*uncer*). The lower part shows the comparison with the SOTA *semi*-supervised methods using stereo as well as depth supervision. Our method outperforms Monodepth2 on all metrics and can also deliver comparable performance to the SOTA semi-supervised method DVSO [55] that additionally uses the depth from Stereo DSO [101] as sparse supervision signal. . . . 139

8.2 Evaluation results of *V2_01* in EuRoC MAV [70]. The performance of monocular depth estimation is boosted largely by the proposed predictive brightness transformation parameters. 141

8.3 Evaluation results of *V2_01* in EuRoC MAV [70] with the model trained with all *MH* sequences. 141

8.4 Results on our test split of KITTI Odometry. The results of the SOTA monocular (M) methods are shown as baselines. The comparison with the SOTA stereo (S) methods shows that D3VO achieves better average performance than other methods, while being a monocular VO. We also show the ablation study for the integration of deep depth(Dd), pose(Dp) as well as uncertainty(Du). 142

8.5 Comparison to other hybrid methods as well as end-to-end methods on Seq.09 and 10 of KITTI Odometry. 142

8.6 Evaluation results on EuRoC MAV [70]. We show the results of DSO and ORB-SLAM as baselines and compare D3VO with other SOTA monocular VIO (M+I) and stereo VIO (S+I) methods. Note that for stereo methods, *V2_03_difficult* is excluded due to many missing images from one of the cameras [34]. Despite being a monocular method, D3VO shows comparable results to SOTA monocular/stereo VIO. The best results among the monocular methods are shown as **black bold** and the best among the stereo methods are shown as **blue bold**. The ablation study shows that *Dd+Dp* delivers large improvement on *V1_03_difficult* and *V2_03_difficult* where the camera motions are very strong. 145

Own Publications

[1] L. von Stumberg, V. Usenko, J. Engel, J. Stueckler, and D. Cremers, "From Monocular SLAM to Autonomous Drone Exploration", in *European Conference on Mobile Robots (ECMR)*, Sep. 2017. DOI: 10.1109/ECMR.2017.8098709. arXiv: 1609.07835 (cit. on pp. 20, 24).

[2] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-Time Trajectory Replanning for MAVs using Uniform B-splines and a 3D Circular Buffer", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, Sep. 2017. DOI: 10.1109/IROS.2017.8202160. arXiv: 1703.01416 (cit. on pp. 20, 24).

[3] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, "MonoRec: Semi-Supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. DOI: 10.1109/CVPR46437.2021.00605. arXiv: 2011.11814 (cit. on pp. 21, 25).

[4] L. von Stumberg and D. Cremers, "DM-VIO: Delayed Marginalization Visual-Inertial Odometry", *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 7, no. 2, pp. 1408–1415, 2022. DOI: 10.1109/LRA.2021.3140129. arXiv: 2201.04114 (cit. on pp. 23, 25, 26, 45).

[5] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers, "GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization", *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, vol. 5, no. 2, pp. 890–897, 2020. DOI: 10.1109/LRA.2020.2965031. arXiv: 1904.11932 (cit. on pp. 23, 24, 29, 65, 88, 89, 94–99, 101).

[6] L. von Stumberg, P. Wenzel, N. Yang, and D. Cremers, "LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization", in *International Conference on 3D Vision (3DV)*, 2020, pp. 968–977. DOI: 10.1109/3DV50981.2020.00107. arXiv: 2010.06323 (cit. on pp. 23, 25, 29, 85).

[7] L. von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization", in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2510–2517. DOI: 10.1109/ICRA.2018.8462905. arXiv: 1804.05625 (cit. on pp. 23, 24, 26, 47, 49, 51, 52, 58, 59, 61, 107, 128, 130, 143, 145).

[8] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1278–1289. DOI: 10.1109/CVPR42600.2020.00136. arXiv: 2003.01060 (cit. on pp. 23, 25, 28, 89, 127).

[9] H. Matsuki, L. von Stumberg, V. Usenko, J. Stueckler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras", *IEEE Robotics and Automation Letters (RA-L) & International Conference on Intelligent Robots and Systems (IROS)*, 2018. DOI: 10.1109/LRA.2018.2855443. arXiv: 1808.02775 (cit. on p. 24).

[10] L. von Stumberg, V. Usenko, and D. Cremers, "A Review and Quantitative Evaluation of Direct Visual–Inertial Odometry," in M. Yang, B. Rosenhahn, and V. Murino, Eds. Academic Press, 2019, ch. Multimodal Scene Understanding, pp. 159–198, ISBN: 978-0-12-817358-9. DOI: 10.1016/B978-0-12-817358-9.00013-5 (cit. on p. 24).

[11] D. Schubert, N. Demmel, L. von Stumberg, V. Usenko, and D. Cremers, "Rolling-Shutter Modelling for Visual-Inertial Odometry", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019. DOI: 10.1109/IROS40897.2019.8968539. arXiv: 1911.01015 (cit. on p. 24).

[12] V. Usenko, L. von Stumberg, J. Stückler, and D. Cremers, "TUM Flyers: Vision—Based MAV Navigation for Systematic Inspection of Structures," in F. Caccavale, C. Ott, B. Winkler, and Z. Taylor, Eds. Cham: Springer International Publishing, 2020, ch. Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users, pp. 189–209, ISBN: 978-3-030-34507-5. DOI: 10.1007/978-3-030-34507-5_8 (cit. on p. 25).

[13] P. Wenzel, R. Wang, N. Yang, Q. Cheng, Q. Khan, L. von Stumberg, N. Zeller, and D. Cremers, "4Seasons: A Cross-Season Dataset for Multi-Weather SLAM in Autonomous Driving", in *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020. DOI: 10.1007/978-3-030-71278-5_29. arXiv: 2009.06364 (cit. on pp. 25, 48, 58, 61).

Bibliography

[14] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Jun. 2004, pp. 652–659. DOI: 10.1109/CVPR.2004.1315094 (cit. on pp. 16, 48, 110).

[15] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 3, 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2658577 (cit. on pp. 16, 18, 26, 38, 49–52, 58, 66, 69, 72, 74, 86, 89–91, 96, 108, 110, 111, 113, 116, 120, 121, 128–131, 134, 136, 137, 142, 143, 145).

[16] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 298–304 (cit. on pp. 16–18, 49, 59, 110, 120, 121, 145).

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014 (cit. on pp. 16, 18, 110, 137).

[18] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera", in *IEEE International Conference on Computer Vision (ICCV)*, 2013 (cit. on pp. 16, 18, 38, 129, 137).

[19] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry", in *arXiv 2016*, 2016 (cit. on pp. 16, 68, 75).

[20] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018 (cit. on pp. 16, 48, 58, 60, 62).

[21] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM", *Robotics: Science and Systems Conference (RSS)*, vol. 2, 2010 (cit. on pp. 16, 128).

[22] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106 (cit. on pp. 16, 86, 89, 130).

[23] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: direct sparse odometry with loop closure", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2198–2204 (cit. on p. 16).

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system", *IEEE Transactions on Robotics (TR-O)*, vol. 31, no. 5, pp. 1147–1163, 2015 (cit. on pp. 16–18, 48, 66, 68, 86, 89, 110, 126, 128, 142, 143, 145).

[25] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces", in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234 (cit. on pp. 16–18, 48, 66, 68, 86, 89, 110).

[26] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "Maplab: An open framework for research in visual-inertial mapping and localization", *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1418–1425, 2018 (cit. on p. 16).

[27] A. Kasyanov, F. Engelmann, J. Stückler, and B. Leibe, "Keyframe-Based Visual-Inertial Online SLAM with Relocalization", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017 (cit. on pp. 16, 121–123).

[28] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018 (cit. on pp. 16, 67, 70, 75, 89).

[29] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007 (cit. on pp. 17, 18, 48, 68, 86, 89, 110).

[30] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time", in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327 (cit. on pp. 17, 18, 49, 69, 110).

[31] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras", *IEEE Transactions on Robotics (TR-O)*, vol. 33, no. 5, pp. 1255–1262, 2017 (cit. on pp. 17, 70, 78, 86, 89, 128, 132, 142).

[32] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM", in *European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849 (cit. on pp. 17, 18, 49, 66, 69, 86, 89, 110, 128, 132, 137).

[33] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM-learning a compact, optimisable representation for dense visual SLAM", *arXiv preprint arXiv:1804.00874*, 2018 (cit. on pp. 17, 131, 152).

[34] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery", *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 422–429, 2020. DOI: 10.1109/LRA.2019.2961227 (cit. on pp. 17, 18, 49, 58, 59, 143, 145).

[35] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020 (cit. on pp. 17, 49, 59).

[36] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator", *IEEE Transactions on Robotics (TR-O)*, vol. 34, no. 4, pp. 1004–1020, 2018 (cit. on pp. 17, 18, 46, 49, 59, 60, 128, 143).

[37] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 796–803, 2017 (cit. on pp. 17, 26, 46, 49, 110, 111, 113, 114, 121, 122, 124, 125, 128, 145).

[38] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap slam", *IEEE Transactions on Robotics (TR-O)*, pp. 1–17, 2021. DOI: 10.1109/TRO.2021.3075644 (cit. on pp. 17, 46, 49, 57, 61).

[39] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2007, pp. 3565–3572 (cit. on pp. 17, 18, 46, 49, 59, 145).

[40] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization", *International Journal of Robotics Research (IJRR)*, vol. 34, no. 3, pp. 314–334, 2014 (cit. on pp. 17, 18, 49, 59, 110, 111, 121–123, 128, 143, 145).

[41] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras", in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016 (cit. on pp. 17, 110–112, 121, 122, 124).

[42] A. Martinelli, "Closed-form solution of visual-inertial structure from motion", *International Journal of Computer Vision (IJCV)*, vol. 106, no. 2, pp. 138–152, 2014, ISSN: 1573-1405. DOI: 10.1007/s11263-013-0647-7 (cit. on pp. 17, 49, 111, 113, 122, 130).

[43] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 1, 2017, ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2521413 (cit. on pp. 17, 46, 49, 111, 122).

[44] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time", in *Robotics: Science and Systems Conference (RSS)*, 2014 (cit. on p. 17).

[45] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 44, no. 01, pp. 154–180, Jan. 2022, ISSN: 1939-3539 (cit. on p. 17).

[46] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013 (cit. on pp. 18, 49, 108, 110).

[47] H. Alismail, B. Browning, and S. Lucey, "Photometric Bundle Adjustment for Vision-Based SLAM", in *Asian Conference on Computer Vision (ACCV)*, 2017 (cit. on pp. 18, 66, 69, 86).

[48] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct sparse mapping", *IEEE Transactions on Robotics (TR-O)*, vol. 36, no. 4, pp. 1363–1370, 2020 (cit. on p. 18).

[49] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "DeMon: Depth and motion network for learning monocular stereo", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5038–5047 (cit. on pp. 19, 131).

[50] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2043–2050 (cit. on pp. 19, 131).

[51] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular visual odometry through unsupervised deep learning", *arXiv preprint arXiv:1709.06841*, 2017 (cit. on pp. 19, 131, 142).

[52] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 7 (cit. on pp. 19, 128, 131, 132, 138, 142).

[53] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 340–349. DOI: 10.1109/CVPR.2018.00043 (cit. on pp. 19, 131, 142).

[54] J. Czarnowski, S. Leutenegger, and A. J. Davison, "Semantic Texture for Robust Dense Tracking", in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017 (cit. on pp. 19, 69, 72).

[55] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry", in *European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833 (cit. on pp. 19, 28, 128, 130, 131, 137, 139, 141–143).

[56] R. Gomez-Ojeda, Z. Zhang, J. Gonzalez-Jimenez, and D. Scaramuzza, "Learning-based image enhancement for visual odometry in challenging hdr environments", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018 (cit. on pp. 19, 69).

[57] L. von Stumberg, "Autonomous flight of a low-cost quadrocopter using a semi-dense monocular slam system," Bachelor's Thesis, Technical University of Munich, 2015 (cit. on p. 20).

[58] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments", *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011 (cit. on p. 20).

[59] L. Koestler, N. Yang, N. Zeller, and D. Cremers, "TANDEM: tracking and dense mapping in real-time using deep multi-view stereo", in *Conference on Robot Learning (CoRL)*, 2021 (cit. on p. 21).

[60] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-D vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26 (cit. on pp. 31, 34).

[61] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment — a modern synthesis", in *Vision Algorithms: Theory and Practice, LNCS*, 2000 (cit. on pp. 31, 32).

[62] F. Dellaert and M. Kaess, "Factor graphs for robot perception", *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017, ISSN: 1935-8253. DOI: 10.1561/2300000043 (cit. on pp. 32, 36).

[63] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944 (cit. on pp. 33, 88, 90).

[64] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963 (cit. on pp. 33, 88, 91).

[65] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers", in *Conference on Computer and Robot Vision (CRV)*, 2015. doi: 10.1109/CRV.2015.52 (cit. on pp. 34, 50).

[66] E. Eade, "Lie groups for 2d and 3d transformations," Tech. Rep., 2017, http://ethaneade.com/lie.pdf (cit. on p. 34).

[67] B. J. Frey, F. R. Kschischang, H.-A. Loeliger, and N. Wiberg, "Factor graphs and algorithms", in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, Citeseer, vol. 35, 1997, pp. 666–680 (cit. on p. 35).

[68] F. Daellert *et al.*, *Gtsam*, https://gtsam.org (cit. on pp. 35, 55).

[69] G. Huang, A. I. Mourikis, and S. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency", in *Int. Symposium on Experimental Robotics (ISER)*, 2008 (cit. on pp. 38, 52).

[70] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets", *International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, 2016. doi: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html (cit. on pp. 48, 58, 68, 75, 82, 109, 121, 131, 134, 135, 138, 140, 141, 145).

[71] E. Hong and J. Lim, "Visual-inertial odometry with robust initialization and online scale estimation", *Sensors*, vol. 18, p. 4287, Dec. 2018. doi: 10.3390/s18124287 (cit. on pp. 49, 59).

[72] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation", in *Robotics: Science and Systems Conference (RSS)*, 2015 (cit. on pp. 50, 51, 110–112).

[73] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions", *IEEE Transactions on Robotics (TR-O)*, vol. 28, no. 1, pp. 61–76, 2012. doi: 10.1109/TRO.2011.2170332 (cit. on pp. 51, 112).

[74] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. doi: 10.1109/ICRA.2014.6907483 (cit. on pp. 51, 112).

[75] C. Campos, J. Montiel, and J. D. Tardós, "Inertial-only optimization for visual-inertial initialization", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 51–57, 2020 (cit. on p. 57).

[76] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2502–2509. doi: 10.1109/ICRA.2018.8460664 (cit. on pp. 59, 143).

[77] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks", in *Advances in neural information processing systems*, 2017 (cit. on p. 66).

[78] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017 (cit. on p. 66).

[79] H. Porav, W. Maddern, and P. Newman, "Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018 (cit. on p. 66).

[80] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cit. on pp. 68, 89, 100, 151).

[81] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator", in *Conference on Robot Learning (CoRL)*, 2017 (cit. on pp. 68, 88, 95).

[82] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", *International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, 2017 (cit. on pp. 68, 88, 95).

[83] S. Park, T. Schöps, and M. Pollefeys, "Illumination Change Robustness in Direct Visual SLAM", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017 (cit. on p. 69).

[84] H. Alismail, M. Kaess, B. Browning, and S. Lucey, "Direct Visual Odometry in Low Light Using Binary Descriptors", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, 2017 (cit. on pp. 69, 72, 89).

[85] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal Correspondence Network", in *Advances in neural information processing systems*, 2016 (cit. on p. 69).

[86] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised Visual Descriptor Learning for Dense Correspondence", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, 2017 (cit. on p. 69).

[87] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks", in *NeurIPS*, 2018 (cit. on p. 69).

[88] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation", *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015 (cit. on p. 69).

[89] C.-H. Chang, C.-N. Chou, and E. Y. Chang, "CLKN: Cascaded Lucas-Kanade Networks for Image Alignment", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2213–2221 (cit. on pp. 69, 89).

[90] C. Tang and P. Tan, "BA-Net: Dense Bundle Adjustment Network", in *International Conference on Learning Representations (ICLR)*, 2019 (cit. on p. 69).

[91] L. Han, M. Ji, L. Fang, and M. Niessner, "RegNet: Learning the optimization of direct image-to-image pose registration", in *arXiv 2018*, 2018 (cit. on p. 69).

[92] Z. Lv, F. Dellaert, J. Rehg, and A. Geiger, "Taking a deeper look at the inverse compositional algorithm", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4581–4590 (cit. on pp. 69, 78, 89).

[93] C. Jaramillo, Y. Taguchi, and C. Feng, "Direct multichannel tracking", in *International Conference on 3D Vision (3DV)*, 2017 (cit. on p. 70).

[94] M. Cummins and P. Newman, "FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance", *International Journal of Robotics Research (IJRR)*, vol. 27, no. 6, 2008 (cit. on pp. 70, 86).

[95] I. Kapsouras and N. Nikolaidis, "A vector of locally aggregated descriptors framework for action recognition on motion capture data", in *EUSIPCO*, 2018 (cit. on p. 70).

[96] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d", in *SIGGRAPH*, 2006 (cit. on p. 70).

[97] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation", in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241 (cit. on pp. 71, 136).

[98] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in *IJCAI*, 1981 (cit. on p. 72).

[99] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization", in *IEEE International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 75).

[100] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing", *arXiv 2018*, 2018 (cit. on p. 76).

[101] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras", in *International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017 (cit. on pp. 77, 96, 100, 128, 130, 132, 134, 139, 142).

[102] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 (cit. on pp. 78, 89, 96, 98, 99).

[103] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236 (cit. on pp. 78, 89, 96, 98, 99, 128).

[104] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2040–2047 (cit. on p. 86).

[105] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 38, no. 4, pp. 652–665, 2015 (cit. on p. 86).

[106] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, *et al.*, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality", *Proceedings of the IEEE*, vol. 106, no. 11, pp. 2020–2039, 2018 (cit. on p. 86).

[107] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis, "Photometric patch-based visual-inertial odometry", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3264–3271 (cit. on p. 86).

[108] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias and rolling shutter effect", *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 2878–2885, 4 Oct. 2018. DOI: 10.1109/LRA.2018.2846813 (cit. on pp. 86, 89, 128).

[109] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6148–6157 (cit. on pp. 88, 89, 94).

[110] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman, "Nid-slam: Robust monocular slam using normalised information distance", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1435–1444 (cit. on p. 89).

[111] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015 (cit. on p. 89).

[112] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004 (cit. on p. 89).

[113] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF", in *IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2011, pp. 2564–2571 (cit. on p. 89).

[114] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel, "R2d2: Reliable and repeatable detector and descriptor", in *NeurIPS*, 2019, pp. 12 405–12 415 (cit. on pp. 89, 96, 98, 99).

[115] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4938–4947 (cit. on pp. 89, 96, 98, 99).

[116] P.-E. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena, "Leveraging deep visual descriptors for hierarchical efficient localization", in *Conference on Robot Learning (CoRL)*, 2018 (cit. on p. 89).

[117] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 716–12 725 (cit. on p. 89).

[118] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "Inloc: Indoor visual localization with dense matching and view synthesis", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7199–7209 (cit. on p. 89).

[119] I. Melekhov, A. Tiulpin, T. Sattler, M. Pollefeys, E. Rahtu, and J. Kannala, "Dgc-net: Dense geometric correspondence network", in *WACV*, IEEE, 2019, pp. 1034–1042 (cit. on p. 95).

[120] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 185–194 (cit. on p. 95).

[121] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cit. on p. 96).

[122] L. von Stumberg, "Direct sparse visual-inertial odometry," M.S. thesis, Technical University of Munich, 2017 (cit. on p. 107).

[123] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain", *International Journal of Robotics Research (IJRR)*, 2012 (cit. on p. 108).

[124] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012 (cit. on pp. 108, 131, 135, 138, 141).

[125] A. Comport, E. Malis, and P. Rives, "Accurate quadri-focal tracking for robust 3D visual odometry", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007 (cit. on p. 110).

[126] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011 (cit. on p. 110).

[127] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012 (cit. on p. 110).

[128] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry", *International Journal of Robotics Research (IJRR)*, vol. 32, 2013 (cit. on p. 110).

[129] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99 (cit. on p. 128).

[130] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN", in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969 (cit. on p. 128).

[131] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9404–9413 (cit. on p. 128).

[132] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks", in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766 (cit. on p. 128).

[133] D. Sun, X. Yang, M. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8934–8943. doi: 10.1109/CVPR.2018.00931 (cit. on p. 128).

[134] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform", in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 467–483 (cit. on p. 128).

[135] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A trainable CNN for joint detection and description of local features", *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 (cit. on p. 128).

[136] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction", *arXiv preprint arXiv:1704.03489*, 2017 (cit. on pp. 128, 131, 132).

[137] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5218–5223 (cit. on pp. 128, 142).

[138] H. Zhan, C. S. Weerasekera, J. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" *arXiv preprint arXiv:1909.09803*, 2019 (cit. on pp. 128, 142, 143).

[139] X. Yin, X. Wang, X. Du, and Q. Chen, "Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5870–5878 (cit. on pp. 128, 142).

[140] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency", *arXiv preprint arXiv:1609.03677*, 2016 (cit. on pp. 128, 131, 134, 136, 139).

[141] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation", in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019 (cit. on pp. 128, 131, 132, 138, 139, 141).

[142] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks", in *International Conference on 3D Vision (3DV)*, IEEE, 2016, pp. 239–248 (cit. on pp. 128, 131).

[143] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction", *arXiv preprint arXiv:1702.02706*, 2017 (cit. on pp. 128, 139).

[144] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005 (cit. on p. 129).

[145] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?" In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 2657–2664. DOI: 10.1109/ROBOT.2010.5509636 (cit. on p. 129).

[146] M. Klodt and A. Vedaldi, "Supervising the new with the old: Learning SFM from SFM", in *European Conference on Computer Vision (ECCV)*, 2018, pp. 698–713 (cit. on pp. 130–132, 134–136).

[147] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" In *Advances in neural information processing systems*, 2017, pp. 5574–5584 (cit. on pp. 130–132, 134, 135).

[148] T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle adjusted direct RGB-D SLAM", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 134–144 (cit. on p. 130).

[149] R. Szeliski, "Image alignment and stitching: A tutorial", *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006 (cit. on pp. 130, 137).

[150] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network", in *Advances in neural information processing systems*, 2014, pp. 2366–2374 (cit. on pp. 131, 138, 139).

[151] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1119–1127 (cit. on p. 131).

[152] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture", in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2650–2658 (cit. on p. 131).

[153] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, United States, 2018 (cit. on p. 131).

[154] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018 (cit. on p. 131).

[155] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 5667–5675. DOI: 10.1109/CVPR.2018.00594 (cit. on p. 131).

[156] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 2022–2030. DOI: 10.1109/CVPR.2018.00216 (cit. on p. 131).

[157] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras", in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019 (cit. on pp. 131, 140–143).

[158] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks", in *Advances in neural information processing systems*, 2015, pp. 2017–2025 (cit. on pp. 131, 134).

[159] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018 (cit. on p. 131).

[160] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep tracking and mapping", in *European Conference on Computer Vision (ECCV)*, 2018, pp. 822–838 (cit. on p. 131).

[161] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 1524–1531 (cit. on p. 132).

[162] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580 (cit. on p. 132).

[163] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "GCNv2: Efficient correspondence prediction for real-time slam", *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3505–3512, 2019 (cit. on p. 132).

[164] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Self-improving visual odometry", *arXiv preprint arXiv:1812.03245*, 2018 (cit. on p. 132).

[165] E. Jung, N. Yang, and D. Cremers, "Multi-Frame GAN: Image Enhancement for Stereo Visual Odometry in Low Light", in *Conference on Robot Learning (CoRL)*, 2019 (cit. on p. 132).

[166] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004 (cit. on p. 134).

[167] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942 (cit. on pp. 134, 142).

[168] H. Jin, P. Favaro, and S. Soatto, "Real-time feature tracking and outlier rejection with changes in illumination", in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, IEEE, vol. 1, 2001, pp. 684–689 (cit. on p. 134).

[169] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001 (cit. on p. 137).

[170] H.-A. Loeliger, "An introduction to factor graphs", *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004 (cit. on p. 137).

[171] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223 (cit. on p. 140).

[172] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8001–8008 (cit. on p. 142).

[173] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised scale-consistent depth and ego-motion learning from monocular video", in *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019 (cit. on p. 142).

[174] T. Feng and D. Gu, "SGANVO: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks", *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4431–4437, 2019 (cit. on p. 142).

[175] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors", *arXiv preprint arXiv:1901.03638*, 2019 (cit. on p. 145).

[176] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry", *IEEE Transactions on Robotics (TR-O)*, vol. 33, no. 1, pp. 1–21, 2016 (cit. on p. 145).

[177] B. Petit, R. Guillemard, and V. Gay-Bellile, "Time shifted imu preintegration for temporal calibration in incremental visual-inertial initialization", in *International Conference on 3D Vision (3DV)*, 2020, pp. 171–179 (cit. on p. 151).

[178] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis", in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 405–421 (cit. on p. 152).

[179] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "Imap: Implicit mapping and positioning in real-time", in *IEEE International Conference on Computer Vision (ICCV)*, 2021, pp. 6209–6218 (cit. on p. 152).

[180] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam", in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022 (cit. on p. 152).