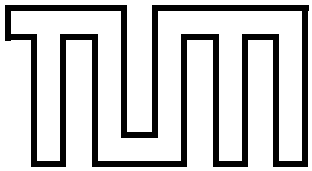
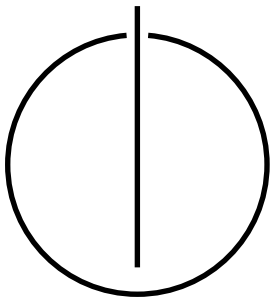


FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Visual-Inertial Navigation for Autonomous Vehicles

Vladyslav Usenko



Technische Universität München

Fakultät für Informatik

Lehrstuhl für Bildverarbeitung und Künstliche Intelligenz

Visual-Inertial Navigation for Autonomous Vehicles

Vladyslav Usenko

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Matthias Nießner
Prüfer der Dissertation:	1. Prof. Dr. Daniel Cremers
	2. Prof. Dr. Roland Siegwart ETH Zürich, Schweiz

Die Dissertation wurde am 16. Oktober 2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 27. Januar 2019 angenommen.

Abstract

Autonomous navigation for vehicles is a fascinating research topic with increasing number of applications in everyday life. To achieve a high level of autonomy in different environments vehicles must rely on onboard sensors. Moreover it is often required that those sensors are low-cost, small, lightweight and low-power. Cameras and inertial measurement units (IMUs) are two sensors that fit those requirements particularly well. In this thesis we explore different aspects of visual-inertial navigation: sensor modeling, odometry, real-time mapping of the environment around the vehicle and using the acquired map for obstacle-free trajectory generation.

First, we focus on the visual-inertial odometry (VIO), which exploits the complementarity of cameras and IMUs to achieve robust and accurate pose estimation. We present two novel methods for visual-inertial odometry that form major contributions of this thesis. The novelty of the methods is that the terms from direct image alignment are combined with IMU terms in a tightly coupled way, which produces better results than losely coupled systems. The first system is based on LSD-SLAM and has an alternating optimization to track the camera and estimate the geometry of the scene. The second system is based on DSO and simultaneously optimizes geometry, pose, velocity, IMU biases and scale. To facilitate thorough evaluation of the VIO systems we have also collected a visual-inertial dataset that provides accurate geometric and photometric calibration.

Estimating the pose of the vehicle is only one part of the autonomous navigation problem. Another part is being able to model the environment around the vehicle and plan the actions according to the current state. A further contribution of this thesis is a novel method for real-time trajectory replanning that addresses this problem. Unlike other camera-based methods that aim to achieve photorealistic reconstruction or large-scale occupancy maps we aim to achieve the highest possible update rate when modeling the environment. We propose to use a vehicle-centered volume represented as a circular buffer that achieves an order-of-magnitude faster measurement insertion time compared to octree-based solutions. For representing the trajectory we use uniform B-splines that ensure the required smoothness of the trajectory and allow for efficient optimization. We formulate an error function that computes an optimal trajectory that follows a global path and penalizes proximity to obstacles based on current estimate of the environment in real time.

Finally, the combination of VIO and real time trajectory replanning for autonomous navigation is demonstrated with a micro aerial vehicle as an example.

Zusammenfassung

Autonome Navigation für Fahrzeuge ist ein faszinierendes Forschungsthema mit einer zunehmenden Anzahl von Anwendungen im Alltag. Um einen hohen Grad an Autonomie in verschiedenen Umgebungen zu erreichen, muss sich das Fahrzeug auf Onboard-Sensoren verlassen, die darüber hinaus oft kostengünstig, klein, leicht und stromsparend sein sollten. Kameras und Inertialsensoren (Intertial Measurement Units – IMUs) sind zwei Sensortypen, die diese Anforderungen besonders gut erfüllen. In dieser Arbeit untersuchen wir verschiedene Aspekte der visuell-inertialen Navigation: Sensormodellierung, Odometrie, Echtzeit-Kartierung der Umgebung des Fahrzeugs und die Verwendung der gewonnenen Karten für die Erzeugung hindernisfreier Trajektorien.

Zunächst konzentrieren wir uns auf die visuell-inertiale Odometrie (VIO), die sich auf die komplementäre Natur von Kameras und IMUs stützt, um eine robuste und genaue Posenschätzung zu erzielen. Wir stellen zwei neue Methoden zur visuell-inertialen Odometrie vor, welche wesentliche Beiträge dieser Doktorarbeit darstellen. Die Neuheit der Methoden ist, dass Fehlerterme aus der direkten Bildregistrierung mit IMU-Fehlertermen auf eng gekoppelte Weise kombiniert werden, was bessere Ergebnisse liefert, als lose gekoppelte Systeme. Das erste System basiert auf LSD-SLAM und beinhaltet eine alternierende Optimierung, um die Kamerapose zu verfolgen und die Umgebungsgeometrie zu schätzen. Das zweite System basiert auf DSO und optimiert gleichzeitig Geometrie, Pose, Geschwindigkeit, sowie systematische IMU-Messabweichung und Skalierung. Um die gründliche Bewertung dieser VIO-Systeme zu erleichtern, haben wir auch einen visuell-inertialen Datensatz aufgenommen, der genaue geometrische und photometrische Kalibrierung bietet.

Die Schätzung der Pose des Fahrzeugs ist nur ein Teil des autonomen Navigationsproblems. Ein anderer Aspekt ist die Modellierung der Umgebung und, basierend auf dem aktuellen Stand, das Planen von Aktionen. Ein weiterer Beitrag dieser Arbeit ist eine neuartige Methode zur Planung von Trajektorien in Echtzeit, die dieses Problem adressiert. Im Gegensatz zu anderen kamerabasierten Methoden, die auf die Erstellung fotorealistischer Rekonstruktionen oder großflächiger Belegungskarten abzielen, wollen wir eine möglichst hohe Updaterate beim Modellieren der Umgebung erreichen. Wir schlagen vor, ein Fahrzeug-zentriertes und als Ringpuffer dargestelltes Volumen zu verwenden, was im Vergleich zu Octree-basierten Lösungen ein um Größenordnungen schnelleres Einfügen von Messdaten erlaubt. Trajektorien werden durch B-Splines dargestellt, die zu ausreichend glatten Trajektorien führen und eine effiziente Optimierung erlauben. Wir stellen eine Fehlerfunktion auf, die in Echtzeit

eine optimale Trajektorie berechnet, welche gleichzeitig einem globalen Pfad folgt und die Nähe zu Hindernissen, basierend auf der aktuellen Umgebungsschätzung, bestraft.

Schließlich demonstrieren wir die Kombination aus VIO und Echtzeit-Trajektorienplanung für die autonome Navigation am Beispiel einer Mikrodrohne.

Acknowledgements

Here, I would like to thank people who helped, encouraged, supported and motivated me during my PhD. First, my doctoral advisor Prof. Daniel Cremers who gave me opportunity to pursue a PhD at the chair and supported me during the program. He managed to build a great team of researchers and I was lucky to work with them and learn from them.

Prof. Roland Siegwart and Prof. Matthias Nießner who agreed to serve as committee members for this dissertation. Sabine Wagner for organizing the paperwork and Quirin Lohr for IT support during the time of my studies.

Jürgen Sturm for his supervision in the beginning of my PhD. Jörg Stückler for his advice and support during the most of my studies and, in particular, with the EuRoC project.

My friends and colleagues from the group, for all our research and social activities. Mariano Jaimez for introducing me to karaoke, Robert Maier for the movies watched together, David Schubert and Linda Martini for making pizza, Christiane Sommer for awesome trips and crazy swimming, Rui Wang for hiking, Csaba Domokos for ice-skating, Lingni Ma for motivation, Mohamed Souiai for making me interested in Forró, Nikolaus Demmel for being a great office mate and co-author, Lukas von Stumberg for his help with EuRoC and collaboration on many other projects. John Chiotellis, Virginia Estellers, Marvin Eisenberger, Thomas Frerix, Vladimir Golkov, Caner Hazırbaş, Christian Kerl, Zorah Lähner, Emanuel Laude, Tim Meinhardt, Laura Leal-Taixé, Thomas Möllenhoff, Rudolph Triebel, Matthias Vestner, Thomas Windheuser, Tao Wu, Nan Yang for making my time at the group an amazing experience.

Surreal team in Oculus Research for having me as an intern, in particular Jakob Engel and Raúl Mur-Artal for helpful discussions.

Finally, I would like to thank my family for their support. My father, who encouraged me to pursue a PhD, my mother, who always supported me on my way and my brother who served me as an example.

Contents

I Introduction and Fundamentals **1**

1 Introduction **3**
1.1 Related Work . . . 4
1.1.1 Structure from Motion . . . 4
1.1.2 Real-Time Visual Odometry and SLAM . . . 5
1.1.3 Visual-Inertial Odometry . . . 7
1.1.4 Environment Representation . . . 7
1.1.5 Optimal Path Planning . . . 9
1.2 Applications . . . 10

2 Contributions and Outline **13**
2.1 Major Contributions . . . 13
2.1.1 Novel Camera Model for Wide-Angle Lenses . . . 15
2.1.2 Direct Semi-Dense Visual-Inertial Odometry for Stereo Cameras 15
2.1.3 Direct Sparse Visual-Inertial Odometry . . . 16
2.1.4 Visual-Inertial Dataset with Precise Geometric and Photometric Calibration . . . 16
2.1.5 Real-Time Trajectory Replanning for MAVs . . . 17
2.2 Outline of the Thesis . . . 18

3 Fundamentals **21**
3.1 3D Geometry . . . 21
3.1.1 Rotation Representations in 3D Space . . . 21
3.1.2 $\mathbf{SO}(3)$ Lie Group and $\mathfrak{so}(3)$ Lie Algebra . . . 23
3.1.3 Rigid Transformations in 3D space . . . 24
3.1.4 $\mathbf{SE}(3)$ Lie Group and $\mathfrak{se}(3)$ Lie Algebra . . . 25
3.2 Probability Theory and Optimization . . . 27
3.2.1 Multivariate Gaussian . . . 27
3.2.2 Maximum Likelihood Estimation . . . 27
3.2.3 Partial Marginalization . . . 28
3.2.4 Maximum Likelihood Estimate for Observations with Gaussian Noise . . . 29

3.2.5 Gauss-Newton Method for Least-Squares Problems 30
3.2.6 Gauss-Newton Method for Smooth Manifolds 31
3.2.7 Levenberg-Marquardt Method 32
3.2.8 Huber Norm and Iteratively Reweighted Least-Squares Problem 32

II Own Publications 35

4 The Double Sphere Camera Model 37
4.1 Introduction 38
4.2 Related Work 39
4.2.1 Pinhole Camera Model 41
4.2.2 Unified Camera Model 42
4.2.3 Extended Unified Camera Model 43
4.2.4 Kannala-Brandt Camera Model 44
4.2.5 Field-of-View Camera Model 46
4.3 Double Sphere Camera Model 47
4.4 Calibration 49
4.5 Evaluation 51
4.6 Conclusion 54

5 Direct Visual-Inertial Odometry with Stereo Cameras 57
5.1 Introduction 58
5.2 Related Work 60
5.3 Contribution. 61
5.4 Notation 61
5.5 Direct Visual-Inertial Stereo Odometry 61
5.5.1 Direct Semi-Dense Stereo Odometry 63
5.5.2 IMU Integration 66
5.5.3 Optimization 67
5.5.4 Partial Marginalization 68
5.5.5 Changing the Linearization Point 70
5.5.6 Statistical Consistency 70
5.6 Results 72
5.6.1 EuRoC Dataset 72
5.6.2 Long-Term Drift Evaluation 73
5.6.3 Malaga Dataset: Autonomous Driving 73
5.7 Conclusion 75

6 Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization 77
6.1 Introduction 78

6.2 Related work . . . 80
6.3 Direct Sparse Visual-Inertial Odometry . . . 81
6.3.1 Notation . . . 81
6.3.2 Photometric Error . . . 82
6.3.3 Inertial Error . . . 82
6.3.4 IMU Initialization and the problem of observability . . . 83
6.3.5 **SIM**(3)-based Representation of the World . . . 83
6.3.6 Scale-aware Visual-inertial Optimization . . . 84
6.3.7 Coarse Visual-Inertial Tracking . . . 88
6.4 Results . . . 90
6.4.1 Robust Quantitative Evaluation . . . 90
6.4.2 Evaluation of the Initialization . . . 93
6.5 Conclusion . . . 94

7 The TUM VI Benchmark for Evaluating Visual-Inertial Odometry 97
7.1 Introduction . . . 98
7.2 Related Work . . . 101
7.3 Sensor Setup . . . 102
7.3.1 Camera . . . 103
7.3.2 Light Sensor . . . 103
7.3.3 IMU . . . 104
7.3.4 Motion Capture System . . . 105
7.4 Calibration . . . 105
7.4.1 Camera Calibration . . . 105
7.4.2 IMU and Hand-Eye Calibration . . . 105
7.4.3 IMU Noise Parameters . . . 107
7.4.4 Photometric Calibration . . . 107
7.5 Dataset . . . 109
7.5.1 Sequences . . . 109
7.5.2 Format . . . 110
7.6 Evaluation . . . 111
7.6.1 Evaluation Metric . . . 111
7.6.2 Results . . . 111
7.7 Conclusion . . . 113

8 Real-Time Trajectory Replanning for MAVs using Uniform B-splines and a 3D Circular Buffer 117
8.1 Introduction . . . 118
8.2 Related Work . . . 120
8.2.1 Trajectory Generation . . . 120
8.2.2 Environment Representation . . . 120
8.3 Trajectory Representation using Uniform B-Splines . . . 121

8.3.1 Uniform B-Splines . . . 121
8.3.2 Comparison with Polynomial Trajectory Representation . . . 124
8.4 Local Environment Map using 3D Circular Buffer . . . 125
8.4.1 Addressing . . . 125
8.4.2 Measurement Insertion . . . 128
8.4.3 Distance Map Computation . . . 128
8.5 Trajectory Optimization . . . 129
8.5.1 Endpoint Cost Function . . . 129
8.5.2 Collision Cost Function . . . 129
8.5.3 Quadratic Derivative Cost Function . . . 130
8.5.4 Derivative Limit Cost Function . . . 130
8.5.5 Implementation Details . . . 131
8.6 Results . . . 131
8.6.1 Three-Dimensional Circular Buffer Performance . . . 133
8.6.2 Optimization Performance . . . 133
8.6.3 System Simulation . . . 133
8.6.4 Real-World Experiments . . . 134
8.7 Conclusion . . . 135

III Conclusion 137

9 Summary 139

10 Future Research 143

IV Appendix 147

A Multimedia Material 149

B Open-Source Code and Datasets 151

List of Figures 153

Own Publications 155

Bibliography 157

Curriculum vitae 169

Part I

Introduction and Fundamentals

Chapter 1

Introduction

Humans have an extraordinary ability to perceive the environment and interact with it. Despite large variations in illumination, time of the day, season, etc, we are able to describe our motion in 3D space, build a map of the environment and successfully navigate between different places. We can also reason about possible collisions and plan our path to avoid them, even in rapidly changing environments. Adding such capabilities to autonomous vehicles represents a significant scientific and engineering challenge, but opens a huge amount of potential applications, e.g. autonomous driving, goods delivery or inspections with micro aerial vehicles, household robotics and others.

Autonomous vehicle require some representation of the environment for navigation. The type of representation depends on the task of the vehicle and available sensors. For the task of localization it is enough to have a sparse map consisting of landmarks that can be easily detected by the sensors mounted on the vehicle, while for obstacle avoidance and path planning a dense occupancy map is preferred. In most cases vehicles use several representations of the environment for different navigation tasks.

We can subdivide the navigation tasks that need to be solved by the autonomous vehicle into: odometry – incremental tracking of the motion relative to the start frame; localization – the problem of finding the location of the robot in a predefined map; SLAM – a combination of the two previous tasks, where building the map is done simultaneously with tracking the camera motion; motion planning – finding an optimal collision-free path from one point to another.

Even though there exist systems for precise localization that rely on infrastructure (GPS outdoors, motion capture systems indoors), to achieve a human level of autonomy for autonomous agents they have to rely on a similar-to-human set of onboard sensors. Cameras and inertial measurement units (IMUs) are particularly interesting for the navigation task because they are cheap, lightweight, have low power consumption and are widely available on the market. Moreover, you can find a direct correspondence to the human sensors – eyes and vestibular system.

1.1 Related Work

In this section an overview of the relevant related work is provided. First, we discuss off-line camera-based solutions for motion estimation and 3D reconstruction, then we discuss real-time visual and visual-inertial odometry and SLAM methods. After that, we provide an overview of different environment representations that can be used for autonomous navigation and an overview of obstacle-free trajectory planning methods.

1.1.1 Structure from Motion

Estimating the motion of the camera and the 3D structure of the environment is an old problem. First approaches to solve it appeared more than a century ago [70] and relied on manually selected correspondences, while more recent approaches date back to the 1980s [77]. Modern methods for structure from motion (SfM) can process thousands of unordered photographs collected from the internet to perform large-scale 3D reconstruction [42] [12]. Some implementations, such as COLMAP [118], OpenMVG [91] and Bundler [120] are even available open-source.

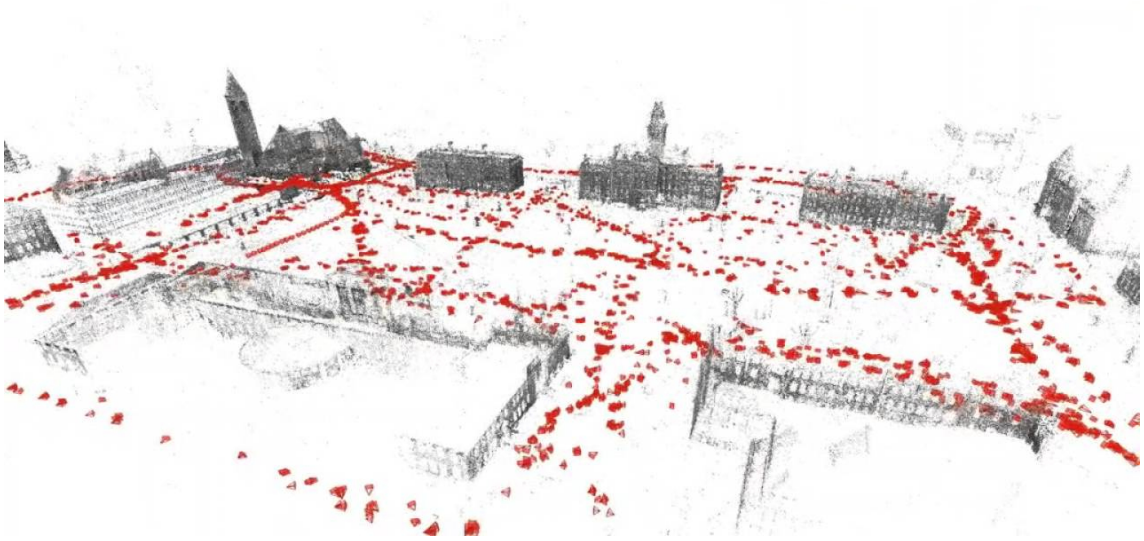


Figure 1.1: Example of a 3D reconstruction with a state-of-the-art SfM system. Illustration taken from [118].

To achieve this level of performance several underlying problems have to be solved. First, we need to identify keypoints that have a unique appearance and can be reliably matched between different images. The keypoint detector proposed

by Harris in 1988 [49] and Shi in 1994 [119] are, despite the age, among the most popular methods even nowadays. For real-time applications several learned keypoint detectors such as FAST [115] and AGAST [83] have been proposed. After detecting keypoints, descriptors such as SIFT [78], SURF [14] and BRIEF [22] can be used to compute keypoint descriptors that are invariant to scale, rotation and affine light transformations. Finally, the descriptors are matched between different images, which allows us to initialize relative pose and camera intrinsic parameters [99]. Initial values are then refined using non-linear optimization methods.

SfM methods can work with unordered image sequences where camera parameters are not known in advance and handle significant variations in illumination due to the time of the day and season changes. All these factors do not allow to achieve real-time performance since slower, but more accurate keypoint detectors and descriptors should be used, and for every new image a time consuming initialization procedure is required.

1.1.2 Real-Time Visual Odometry and SLAM

The task of incrementally tracking the camera motion from video is commonly called visual odometry. The fact that we have a calibrated camera and can assume that the sequence of images is temporally consistent allows us to make visual odometry methods real-time capable.

First attempts to use cameras for incremental motion estimation were done in the 1970s [90]. The term visual odometry was first introduced by Nister et al. [100], who proposed to match a sparse set of keypoints from frame to frame, yielding multiple observations of the same point in different images. The 3D camera motion is then computed by minimizing the reprojection error. Such systems were also successfully deployed on the Mars rovers in NASA's Mars exploration program [82].

The first systems capable of performing visual odometry in real-time were based on the Extended Kalman Filter (EKF) [26, 29]. These methods included both the motion parameters of the camera and 3D landmarks in the state space and formulated filter updates minimizing the reprojection error. In this formulation of the visual odometry problem the computation complexity scales cubicly with the size of the state space, which limits the size of the map. In practice, these systems can be used only in very small environments.

The next generation of algorithms was not only able to track the camera motion, but also reconstruct the map with subsequent refinement of the trajectory [68, 94]. These methods are called SLAM methods, which stands for simultaneous localization and mapping. They are based on bundle adjustment which runs in the background thread and maintains a map consisting of a subset of selected frames (keyframes) and a camera tracking thread that uses the map to track the camera motion for every frame. Overall, these methods are capable of tracking the camera motion and creating a consistent map with the ability to close trajectory loops and relocalize

the camera, outperforming previous methods in accuracy and robustness.

Recently, direct methods for visual odometry became popular. 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 [27], where the authors do not explicitly reconstruct the 3D environment, but formulate quadrifocal constraints on pixels for estimating the motion. The introduction of consumer RGB-D cameras caused a rapid development of direct methods, since depth for every pixel is provided by the camera. This makes it easy to formulate the visual odometry problem. Several methods for motion estimation for RGB-D cameras were developed by Kerl et al. [65, 66] and Newcombe et al. [96].

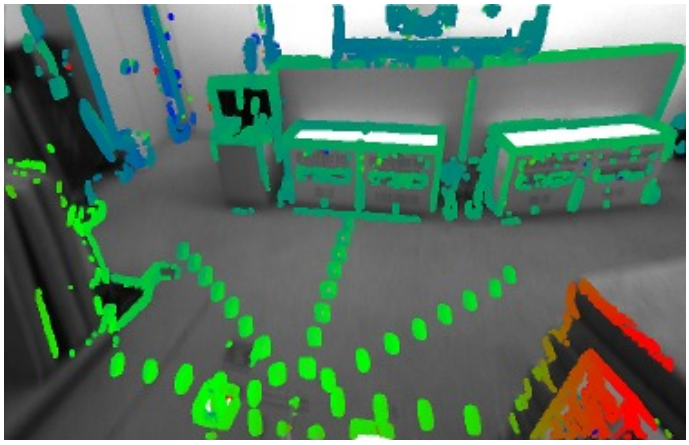


Figure 1.2: An example of the direct visual-inertial odometry method presented in this thesis.

Direct approaches for visual odometry and SLAM can be also used with monocular cameras. There exist several examples of such algorithms that work with dense, semi-dense and sparse maps of the environment. Newcombe et al. [97] proposed a dense approach which reconstructs the depth map for the entire image. To achieve the dense reconstruction a smoothness assumption has to be made, because only the image regions with a sufficient gradient can be used for precise depth estimation. Another approach is to ignore the regions of the image that do not contain enough information for depth estimation. This approach is used by Engel. et al. [33], which is an example of a semi-dense SLAM system.

Direct sparse methods have been discussed in two recent publications. SVO proposed by Forster et al. [40] uses a keypoint-based approach for mapping keyframes and performs sparse direct image alignment to track intermediate frames. DSO proposed by Engel et al. [32] is a purely direct and sparse system. By using a sparse set of points it can simultaneously optimize the 3D structure of the environment and camera motion parameters on a window of keyframes yielding superior accu-

racy and robustness compared to the semi-dense approach, which uses alternating optimization.

1.1.3 Visual-Inertial Odometry

Cameras and IMUs complement each other in many ways. The accelerometer and gyroscope measurements provide good short-term motion information, and the cameras eliminate long-term drift and allow large-scale mapping and localization. Furthermore, an IMU makes scale as well as the roll and pitch angles observable, which is relevant for many applications. Therefore there were several methods which combine an IMU with vision to solve the visual odometry problem. By the type of IMU integration these methods can be classified into two general approaches.

The first one is the so-called *loosely coupled* fusion. In such systems the vision sub-system is used as a blackbox, providing pose measurements, which are then combined with inertial data in an (extended or unscented) Kalman filter. Examples of such systems are presented in [35, 87, 130].

Recently, several *tightly coupled* approaches were presented. In these approaches the motion of the camera is computed by jointly optimizing the parameters in a combined energy function consisting of visual and inertial error terms. This makes the optimization better constrained and better captures correlations between different sensors, which results in better accuracy and robustness. IMU data also helps to better initialize the optimization, which helps for the cases when the energy function has many local minima. Tightly coupled fusion has been presented for filtering-based approaches [17, 74, 126] as well as for energy-minimization-based systems [39, 73, 95, 8].

Since the cost function that we try to minimize is highly non-convex, an initialization close to the global minimum is required for the optimization to converge. In the case of visual-inertial odometry initial velocity, bias and gravity direction have to be initialized. Therefore apart from the usual visual initialization procedure most visual-inertial systems need a specific visual-inertial initialization. However the problem is even more difficult, because several types of motion do not allow to uniquely determine all variables. In [85] a closed-form solution for visual-inertial initialization was presented, as well as an analysis of exceptional cases, which was later extended to account for IMU biases by [60].

1.1.4 Environment Representation

The autonomous navigation problem usually requires different environment representations for different subtasks. Figure 1.3 shows an example of two maps generated from the same sensor data in an indoor GPS-denied environment. The keypoint map is used for vehicle localization and the occupancy map is used for obstacle avoidance and path planning. Since the type of the map used for localization is defined

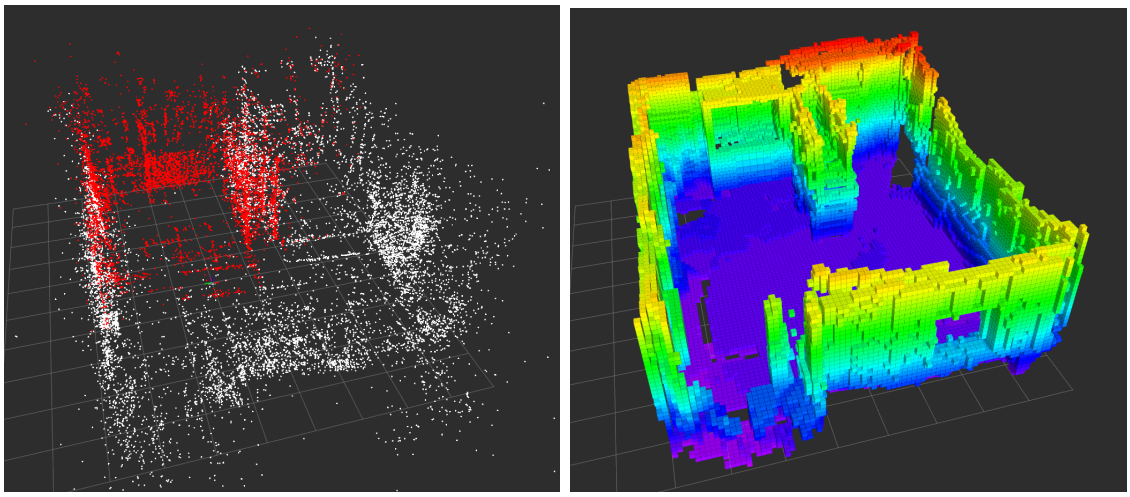


Figure 1.3: Example of representations of an indoor GPS-denied environment used for autonomous vehicle navigation. A keypoint-based map used for localization is shown on the left and an octree-based occupancy map used for path planning and collision avoidance is shown on the right.

by the selection of the visual odometry or SLAM system discussed in the previous subsections, in this subsection we discuss the map representations that can be used for path planning and collision avoidance.

In order to plan a collision-free trajectory, an environment representation that stores information about the occupancy is necessary. In the 3D case the simplest solution that can be used for this purpose is a voxel grid. In this representation, a volume is subdivided into a regular grid of smaller sub-volumes (voxels), where information about the occupancy is stored. The main disadvantage of this approach is its large memory-footprint, which does not allow to map large volumes. However, the advantage is very fast constant-time access to any of the elements.

To cope with the memory limitation, octree-based representations of the environment were presented in [52] and [122]. They store information in an efficient way by pruning the leaves of the octree that contain the same values, but this results in logarithmic access times for each element, instead of constant time as in voxel-based representations.

Another widespread approach to environment mapping is voxel hashing, which was proposed by [98] and used in [102]. It is mainly used for storing a truncated signed distance function representation of the environment. In this case, only a narrow band of measurements around the surface is inserted and only the memory required for that sub-volume is allocated. However, this approach does not offer significant advantages when storing non-truncated measurements or dense information.

1.1.5 Optimal Path Planning

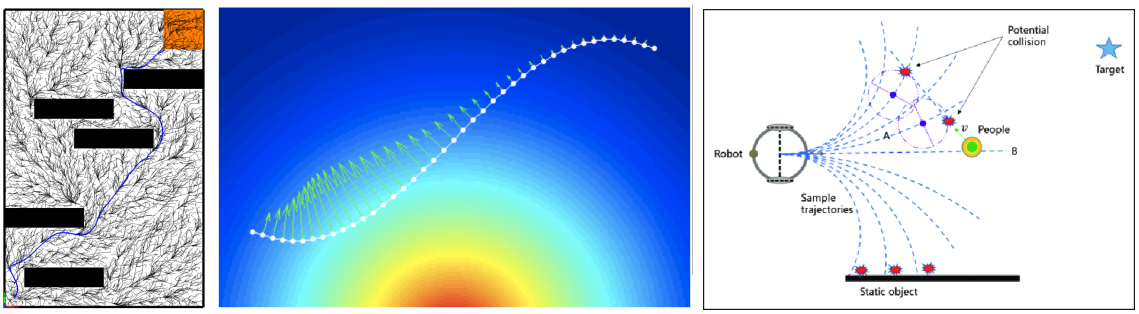


Figure 1.4: Path planning approaches. A search-based path planning approach based on rapidly exploring random tree (RRT) is shown on the left. The optimization-based approach from [133] is shown in the middle. The dynamic window approach, one of the motion-primitive-based approaches, proposed in [41] is shown on the right. Illustrations taken from [25, 113, 133].

There exist three main approaches for trajectory generation: search-based path planning followed by smoothing, optimization-based approaches and motion-primitive-based approaches.

In search-based approaches, first, a non-smooth path is constructed on a graph that represents the environment. The graph can be a fully-connected grid as in [31] and [59], or be computed using a sampling-based planner (rapidly exploring random tree, probabilistic roadmap) as in [113] and [21]. After that, a smooth trajectory represented by a polynomial, B-spline or discrete set of points is computed to closely follow this path. This class of approaches is currently the most popular choice for large-scale path planning problems in cluttered environments where a map is available a priori.

Optimization-based approaches minimize a cost function that consists of smoothness and collision terms. The trajectory in such approaches can be represented as a set of discrete points [133] or polynomial segments [101]. The approach presented in Chapter 8 falls into this category, but represents a trajectory using uniform B-splines.

A different group of approaches is based on path sampling and motion primitives. Motion primitives were successfully applied to a flight through the forest [105] and autonomous robot navigation [41], and sampling-based approaches were successfully used for challenging tasks such as ball juggling [93], but the ability of both approaches to find a feasible trajectory largely depends on the selected discretization scheme.

Figure 1.5: The Standford AI cart from the 1970s shown on the left was one of the first robots to use cameras for navigation [90]. One of Waymo's modern self-driving cars that have autonomously driven 3.5 million miles and have a 360-degree camera system among other sensors [129] is shown on the right. Illustrations taken from [90, 129].

1.2 Applications

Visual-inertial navigation has a large number of applications that potentially have a huge impact on everyday life. Some examples of such applications are described in the following.

Autonomous driving has gone a long path from early prototypes with very limited functionality in the 1970s [90] to autonomous cars that have driven millions of kilometers on public roads [129].

Most of the current prototypes rely on Lidar for localization and obstacle detection due to the advantages of this technology: it can precisely measure the distance to objects that are hundreds of meters away; as an active sensor it works at night and in low light conditions; Lidar measurements are not subject to drastic changes of appearance and work well in rain and fog. However, the complex mechanical structure which degrades over the years, the large price and power consumption motivate the manufacturers to look for alternatives.

Camera-based solutions are one of the alternatives, and in fact are frequently used for redundancy. Vision-based solutions can not only be used for localization, but due to the large amount of information provided by the cameras and recent advances in machine learning it is possible to use cameras for depth estimation [71], vehicle, pedestrian and traffic light detection [111].

Micro aerial vehicles (MAVs) have recently moved from research prototypes to common consumer products. In order to control the position of the MAV, or

to implement trajectory following, position feedback is required. Most consumer MAVs rely on GPS for that purpose, which constrains MAV applications to places where good GPS reception is available.

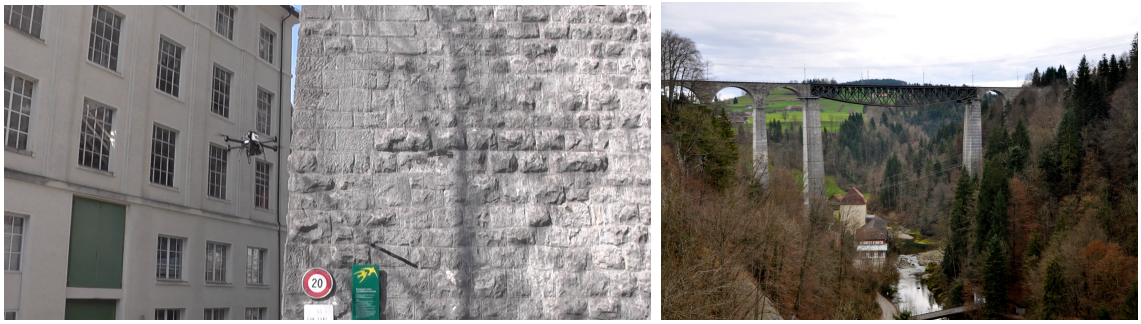


Figure 1.6: An MAV autonomously inspects the structure of a bridge. Since the GPS reception under the bridge is unreliable the MAV relies on visual and inertial sensors to estimate the position in the environment and plan the trajectory.

For applications like package delivery where the MAV has to operate close to buildings at the last stage of the delivery, or for inspection tasks in areas where the GPS signal is reflected or absent (Figure 1.6) it is important to obtain pose feedback from the onboard sensors. Even though sometimes different sensors (such as Lidar, RGB-D cameras, sonars and radars) can be used, using a combination of passive cameras and IMUs is often preferred because of the size, weight and power consumption.

Service Robotics is another application area for visual and visual-inertial navigation. The Dyson 360 Eye vacuum cleaner robot was one of the first consumer products to use vision-based navigation to localize itself in the environment and clean it in a systematic way.

Virtual and Augumented Reality is another application area where precise camera tracking is essential. It is required to correctly render virtual objects in the world and track the precise location of the user. Creating environment maps on-line allows to correctly process occlusions and user interactions with the environment. Since most smartphones are already equipped with cameras and IMUs, this makes visual-inertial odometry and SLAM an obvious choice for this task.

Chapter 2

Contributions and Outline

The purpose of this thesis is to develop the technologies relevant for autonomous navigation of vehicles using onboard sensors, in particular the combination of cameras and IMUs. First, we investigate models of the sensors and present a novel camera model for wide-angle lenses. Then, we present two direct visual-inertial odometry methods (semi-dense and sparse) and provide a dataset for the evaluation of visual-inertial odometry. After that, we propose a novel method for real-time trajectory replanning and demonstrate autonomous navigation of a flying vehicle.

2.1 Major Contributions

This cumulative thesis comprises five full-length publications [4, 5, 7, 8, 10], which are the result of joint work with Lukas von Stumberg, Nikolaus Demmel, David Schubert, Thore Goll, Andrej Pangercic, Jakob Engel, Jörg Stückler and Prof. Daniel Cremers. All these works were published in highly ranked, peer-reviewed international conferences. The paper [5] is based on the Master's thesis of Lukas von Stumberg supervised by me. The paper [10] was finalist of the Best Paper Award at the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Table 2.1 shows a complete summary of all works published as part of, or in conjunction with this thesis. It also lists other publications done while pursuing this degree, originating from different projects, which are not included as part of this cumulative thesis.

In this thesis a novel camera model, two visual-inertial odometry systems, a visual-inertial dataset and a system for real-time trajectory replanning are presented. We also demonstrated how all these components can be used for navigating a vehicle using onboard sensors in an unknown environment, using a micro aerial vehicle as an example. Overall the contributions of the thesis are as follows.

Table 2.1: **Full Publication Summary.** Complete list of publications done while pursuing this degree, ordered chronologically. For publications that are included in this cumulative thesis, we list the respective chapter. Publications that are not part of this thesis are listed in gray.

V. Usenko, J. Engel, J. Stückler and D. Cremers. "Reconstructing Street-Scenes in Real-Time from a Driving Car". In: *2015 International Conference on 3D Vision*. IEEE, Oct. 2015. DOI: 10.1109/3dv.2015.75

V. Usenko, J. Engel, J. Stückler and D. Cremers. "Direct Visual-Inertial Odometry with Stereo Cameras". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. DOI: 10.1109/icra.2016.7487335 (Chapter 5)

J. Engel, V. Usenko and D. Cremers. "A Photometrically Calibrated Benchmark for Monocular Visual Odometry". In: *arXiv preprint arXiv:1607.02555* (2016). eprint: http://arxiv.org/abs/1607.02555v2

L. von Stumberg, V. Usenko, J. Engel, J. Stückler and D. Cremers. "From Monocular SLAM to Autonomous Drone Exploration". In: *2017 European Conference on Mobile Robots (ECMR)*. IEEE, Sept. 2017. DOI: 10.1109/ecmr.2017.8098709. eprint: http://arxiv.org/abs/1609.07835v3

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: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8202160 (Chapter 8)

L. von Stumberg, V. Usenko and D. Cremers. "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. May 2018. DOI: 10.1109/ICRA.2018.8462905. eprint: http://arxiv.org/abs/1804.05625v1 (Chapter 6)

V. Usenko, N. Demmel and D. Cremers. "The Double Sphere Camera Model". In: *Proc. of the Int. Conference on 3D Vision (3DV)*. Sept. 2018. DOI: 10.1109/3DV.2018.00069. eprint: http://arxiv.org/abs/1807.08957 (Chapter 4)

D. Schubert, N. Demmel, V. Usenko, J. Stückler and D. Cremers. "Direct Sparse Odometry With Rolling Shutter". In: *Proc. of the European Conference on Computer Vision (ECCV)*. Sept. 2018. eprint: http://arxiv.org/abs/1808.00558

D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler and D. Cremers. "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robot Systems (IROS)*. Oct. 2018. eprint: http://arxiv.org/abs/1804.06120v1 (Chapter 7)

H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler and D. Cremers. "Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras". In: *IEEE Robotics and Automation Letters* (2018). DOI: 10.1109/lra.2018.2855443. eprint: http://arxiv.org/abs/1808.02775

2.1.1 Novel Camera Model for Wide-Angle Lenses

One contribution of this thesis is a novel camera model for cameras with wide-angle lenses. Recently several researchers have shown the benefit of using cameras with a large field of view for vision-based motion estimation, which results in increased accuracy and robustness.

In Chapter 4 we provide an extensive review of existing camera models for large-field-of-view cameras. To make the thesis self-contained, for each of the models we provide projection and unprojection functions and the subspace of points which result in valid projections. After that, we propose the new Double Sphere camera model that fits well for lenses with a large field of view, has a closed form inverse and is computationally inexpensive. We evaluate the models on a dataset collected with several different wide-angle lenses. The models are compared using the metrics that are relevant for visual odometry: reprojection error, computation time for projection and unprojection functions and their Jacobians. We also provide qualitative results, analyse the performance of all models and discuss the suitability of the models for real-time visual odometry and SLAM applications.

2.1.2 Direct Semi-Dense Visual-Inertial Odometry for Stereo Cameras

In Chapter 5 we propose a novel direct visual-inertial odometry method for stereo cameras based on [33]. In this method pose, linear velocity and IMU biases are estimated by minimizing a combined photometric and inertial energy functional. This allows us to exploit the complementary nature of inertial and visual data. The proposed method belongs to the direct methods. Here geometry is estimated in the form of semi-dense depth maps instead of manually designed sparse keypoints. For depth estimation the method uses both static and temporal stereo, which results in depth maps with metrically correct scale. This property is important since camera tracking and depth estimation are performed it two separate optimization processes, so the IMU data is not able to propagate scale information from the tracking optimization to the estimated depth maps. Because of that, initialization of the depth map with proper scale is essential for the operation of the system. Evaluation showed that our method outperforms not only vision-only or loosely coupled approaches, but also can achieve more accurate results than state-of-the-art keypoint-based methods on different datasets, including sequences with rapid motion and significant illumination changes. In addition, the method provides accurate semi-dense, metric reconstructions of the environment, which can be used for obstacle avoidance and exploration algorithms [6].

2.1.3 Direct Sparse Visual-Inertial Odometry

In Chapter 6 we present a novel approach for visual-inertial odometry based on [32]. It jointly estimates camera poses and sparse geometry of the scene by minimizing photometric and IMU measurement terms in a combined energy functional. This constitutes the main difference to the method described in Chapter 5, since now we can actually propagate scale information from the IMU measurements to the geometry of the scene. This way we can reconstruct metrically correct geometry of the environment with a single monocular camera and an IMU.

In the proposed method the system performs a bundle-adjustment-like optimization on a sparse set of points. Unlike regular bundle adjustment, the system minimizes the photometric error, instead of minimizing the reprojection error of the detected keypoints. Because of that, the system is able to track any pixels with large enough intensity gradients. To cope with the fact that the IMU measurements have a much higher rate than the camera measurements, a preintegration is used to summarize several consecutive measurements into one pseudo-measurement.

Scale and gravity direction are explicitly included into our model and jointly optimized with other variables. Since with a monocular camera the scale is not immediately observable we initialize our visual-inertial system with an arbitrary scale instead of delaying the initialization. To ensure a bounded computation time we marginalize out old states as described in Section 3.2.3. In order to keep the system consistent we introduce a novel strategy which we call dynamic marginalization. This strategy allows us to use partial marginalization even in the cases when scale is initialized far from the true value. Evaluation on the EuRoC dataset shows that the proposed method outperforms the state of the art and the method presented in Chapter 5.

2.1.4 Visual-Inertial Dataset with Precise Geometric and Photometric Calibration

In Chapter 7, we propose the TUM VI benchmark, a novel dataset with 28 sequences in different environments for evaluating visual-inertial odometry. It contains one-megapixel camera images at 20 Hz with a high dynamic range. Moreover, the two cameras have a linear response function and pre-calibrated vignetting which gives additional information about the image formation process. The cameras and the IMU in all sequences are time-synchronized in hardware. The IMU contains a three-axis accelerometer and gyroscope and provides measurements at 200 Hz. Figure 2.1 shows the setup that was used for collecting the dataset.

At the start and end segments of the sequences the ground-truth pose from a motion capture system at 120 Hz is available. To provide the pose information in the IMU frame we performed hand-eye calibration on the calibration sequences. For other sequences we perform time alignment to the motion capture data using

rotational velocities and gyroscope measurements.

The full dataset with raw and calibrated data is publicly available. The main advantage over the existing datasets is the precise geometric and photometric calibration, high resolution, wide field of view of the cameras and high dynamic range of the collected image data.

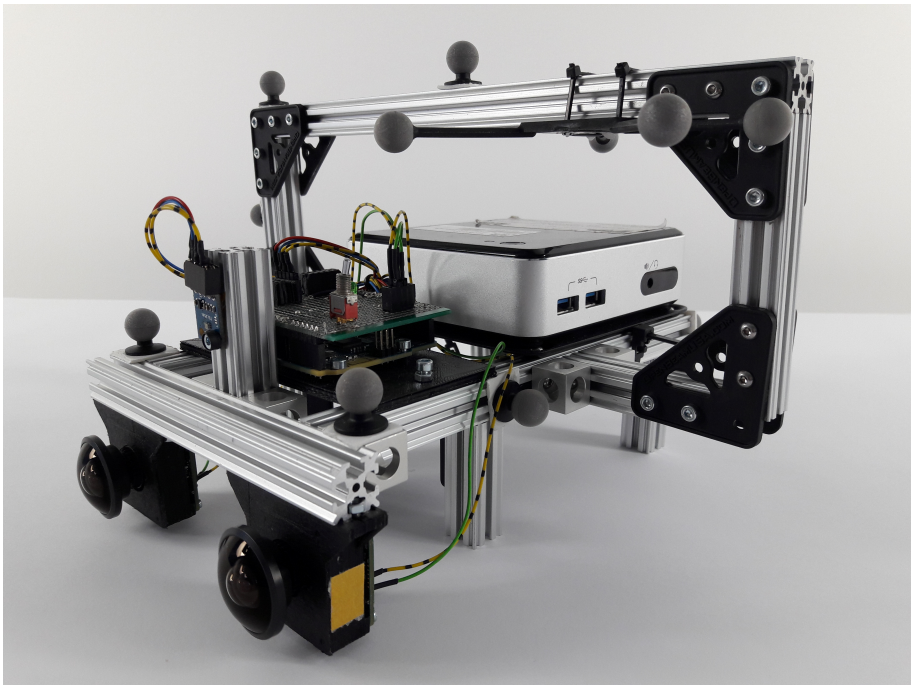


Figure 2.1: Setup that was used for collecting the visual-inertial dataset. Two wide-angle cameras are hardware-synchronized with the IMU to ensure accurate timestamps. Reflective markers mounted on the setup are used to obtain the ground-truth pose from the motion capture system.

2.1.5 Real-Time Trajectory Replanning for MAVs

In Chapter 8, we propose a real-time approach to local trajectory replanning for micro aerial vehicles (MAVs). Trajectory generation for static environments, where the map is known a priori, is a well studied problem with many solutions proposed over the last years. In this thesis, we assume there exists a planned global trajectory and introduce an algorithm that can avoid unmodeled (possibly dynamic) obstacles while closely following the global trajectory in the obstacle-free environment (Figure 2.2).

To make the proposed approach real-time capable, we maintain the information about the environment around the MAV in a robocentric occupancy grid stored in a three-dimensional circular buffer. This gives us a magnitude faster measurement

insertion time compared to competing methods. The trajectory that we optimize is represented with uniform B-splines, which ensures the required smoothness and fast optimization.

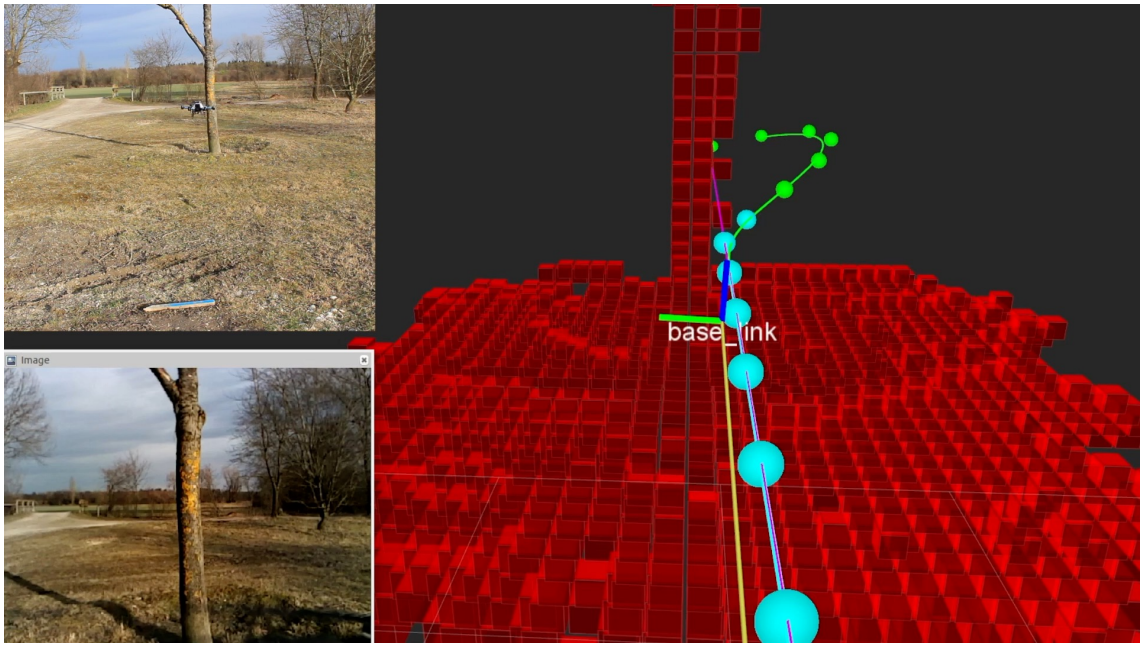


Figure 2.2: Autonomous navigation of a flying vehicle. Visual-inertial odometry is used to track the 3D position of the MAV. Occupancy information is stored in the robocentric 3D circular buffer shown on the right. The trajectory is planned according to the most recent state of the environment with the global path shown in pink and the optimized trajectory shown in green (still being optimized) and blue (fixed). An image from the on-board camera is shown on the bottom left and the corresponding third-person view is shown on the top left.

2.2 Outline of the Thesis

This thesis consists of three parts. Part I introduces the research problem, provides motivation and possible applications, and introduces state-of-the-art methods for visual-inertial odometry, 3D mapping and optimal trajectory generation. Chapter 1 presents the research problem, a review of the related work and applications. Chapter 2 states the contribution of this thesis and provides an overview of selected publications. Chapter 3 gives the necessary mathematical background, which includes 3D geometry, probability theory and non-linear optimization.

Part II provides the main content of the cumulative thesis that consists of five research publications. To achieve high-accuracy localization precise modeling of

the sensors is required. This is addressed in Chapter 4, which introduces a novel camera model for wide-angle and fisheye cameras. Chapter 5 presents a novel direct semi-dense visual-inertial odometry for stereo cameras based on [33]. This tightly coupled method, combines direct image alignment and IMU terms in one energy functional for camera tracking, while the mapping part uses stereo images to recover metrically scaled semi-dense depth maps. Chapter 6 presents an improved version of visual-inertial odometry based on [32]. Unlike the previous version it simultaneously estimates camera motion and 3D geometry in one combined functional, which makes it possible to use monocular cameras and still recover the metric scale from IMU measurements in one combined optimization. To evaluate the methods we need publicly available datasets with time-synchronized IMU and camera measurements. Even though several such datasets exist they do not have photometric calibration. This yields sub-optimal performance for direct methods. Chapter 7 presents a public dataset that provides precise geometric and photometric calibration and ground-truth poses for the start and end of the sequences. Assuming that we can track the 3D position of a vehicle Chapter 8 presents a novel method for optimal trajectory planning based on the current environment state.

Part III concludes the thesis with Chapter 9 which states the limitations of the current approaches and discusses possible future work and remaining challenges in Chapter 10.

Chapter 3

Fundamentals

In this chapter we provide the fundamental mathematical concepts that will be used in the thesis. First, we discuss 3D geometry and pose representations. After that, we describe the foundations of probability theory and nonlinear optimization.

3.1 3D Geometry

3.1.1 Rotation Representations in 3D Space

There exist several ways to represent a rotation in 3D space. A rotation in 3D space has three degrees of freedom but some representations are overparametrized. In this subsection Euler angles, unit quaternions and rotation matrices that form the $\mathbf{SO}(3)$ Lie Group are discussed.

Euler angles are three angles, that represent the orientation of an object as a sequence of rotations around defined axes. Given that the order of the axes around which the rotations are performed can be different, there exist 12 different sequences of rotations. One of the typical sequences is Yaw-Pitch-Roll (YPR). In this notation first a rotation around the Z axis is performed, then around the modified Y axis and finally around the modified X axis for yaw, pitch and roll angles respectively. This representation is minimal, because it uses three parameters to describe three degrees of freedom of the object orientation. However this representation has a degenerate case called Gimbal Lock. If the roll angle is equal to $\pm 90°$ then there is no unique correspondences between object orientation and these three angles, because in this case a change in roll causes a change in yaw. This requires handling these cases with special care. Another issue with Gimbal Lock is that the Jacobian in this configuration is rank deficient which makes it not suitable for optimization. Concatenation of several rotations represented as Euler angles is not straightforward, so it is better to use other rotation representations for this task.

Unit quaternions use 4 parameters to represent an orientation of the body.

$$q = (q_x, q_y, q_z, q_w), \tag{3.1}$$

where q_x, q_y and q_z represent the imaginary part of the quaternion and q_w represents the real part of the quaternion. Here we use Hamilton convention for quaternions, but there exist other notations such as JPL [121]. Essentially, a quaternion represents a rotation for an angle around an axis. If we have an angle α and a unit vector $n = (n_x, n_y, n_z)$ representing the axis of rotation we can construct a quaternion using the following equations

$$q_x = \sin(\frac{\alpha}{2})n_x, \tag{3.2}$$

$$q_y = \sin(\frac{\alpha}{2})n_y, \tag{3.3}$$

$$q_z = \sin(\frac{\alpha}{2})n_z, \tag{3.4}$$

$$q_w = \cos(\frac{\alpha}{2}). \tag{3.5}$$

The corresponding rotation matrix can be computed from a quaternion as follows

$$R = \begin{pmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_zq_w & 2q_xq_z + 2q_yq_w \\ 2q_xq_y + 2q_zq_w & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_xq_w \\ 2q_xq_z - 2q_yq_w & 2q_yq_z + 2q_xq_w & 1 - 2q_x^2 - 2q_y^2 \end{pmatrix}. \tag{3.6}$$

This representation does not have a gimbal lock problem, but the quaternions q and $-q$ represent the same rotation. Concatenation of several rotations represented as quaternions is straightforward and done by quaternion multiplication. Also unit quaternions might de-normalize because of rounding errors, so they have to be periodically normalized. The normalization, however, is simple and computationally inexpensive compared to the normalization of rotation matrices.

Rotation Matrices are 3×3 orthonormal matrices with $|R| = 1$ which represent the orientation of a rigid object.

$$R = \begin{pmatrix} r_{1,1} & r_{1,1} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \tag{3.7}$$

This representation specifies the orientation of the rigid body relative to some external coordinate system. To find the coordinates (x', y', z') of a rotated point we have to simply multiply the rotation matrix with the original point (x, y, z)

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.8}$$

Rotation matrices can be easily concatenated using matrix multiplication. This representation of a rotation is over-parametrized, because it uses 9 values to represent 3 degrees of freedom. The other 6 degrees of freedom are constrained by orthonormality assumption.

3.1.2 $\mathbf{SO}(3)$ Lie Group and $\mathfrak{so}(3)$ Lie Algebra

Rotation matrices form the $\mathbf{SO}(3)$ group under the matrix multiplication operation. This means that when multiplying rotation matrices the result will always be a rotation matrix and that the other group axioms (associativity, neutral element, unique inverse element) hold

$$R_1(R_2R_3) = (R_1R_2)R_3, \tag{3.9}$$

$$R_1I = IR_1 = R_1 \tag{3.10}$$

$$R_1R_1^{-1} = R_1^{-1}R_1 = I, \tag{3.11}$$

where $R_1, R_2, R_3 \in \mathbf{SO}(3)$ and I is the identity matrix. Moreover, for rotation matrices the inverse of the matrix is its transpose $R^{-1} = R^T$.

For Lie groups we can also define a corresponding Lie algebra that describes the tangent space around identity. In this case we define a vector $\xi \in \mathbb{R}^3$ from which we can construct the corresponding element in the $\mathfrak{so}(3)$ lie algebra using the *hat* operator and the corresponding rotation matrix $R \in \mathbf{SO}(3)$ using matrix exponentiation.

$$R = e^{\hat{\xi}}, \tag{3.12}$$

$$\hat{\xi} = \sum_{i=1}^{3} G_i\xi(i), \tag{3.13}$$

where $\xi(i)$ is the i-th component of vector ξ and G_i is the i-th generator of the $\mathbf{SO}(3)$ group

$$G_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, G_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{3.14}$$

and e stands for matrix exponentiation

$$e^X = \sum_{i=0}^{\infty} \frac{1}{i!}X^i. \tag{3.15}$$

In the case of the $\mathbf{SO}(3)$ group a closed-form solution exists, which is called Rodrigues formula

$$e^{\hat{\xi}} = \mathbf{I} + \frac{\sin(||\xi||)}{||\xi||}\hat{\xi} + \frac{1 - \cos(||\xi||)}{||\xi||^2}\hat{\xi}^2. \tag{3.16}$$

The inverse operator, which is called log, also exists in closed form for the $\mathbf{SO}(3)$ group

$$\log(R) = \frac{\arccos(d)}{2\sqrt{1-d^2}}(R - R^T), \tag{3.17}$$

$$d = \frac{1}{2}(\mathrm{tr}(R) - 1), \tag{3.18}$$

where tr is the trace operator. The log operator gives the element in $\mathfrak{so}(3)$, which we can compress into a three dimensional vector $\xi = \log(R)^\vee$ using the operator *vee* which is the inverse of the *hat* operator.

If we unroll the resulting 3×3 matrix to a 9 dimensional vector, the Jacobian of this function at $\xi = 0$ is

$$\left.\frac{\partial e^{\hat{\xi}}}{\partial \xi}\right|_{\xi=0} = \begin{pmatrix} | & | & | \\ G_1 & G_2 & G_3 \\ | & | & | \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.19}$$

This Jacobian is used in the iterative numeric optimization methods discussed in the following sections.

3.1.3 Rigid Transformations in 3D space

Rigid transformations, also referred to as Isometries, are the transformations that preserve the distance between each pair of points and the orientation between each pair of vectors [19]. In three dimensional space rigid transformations are called Euclidean transformations. These transformations include translation, rotation and in some sources reflection, however in this thesis we only consider rigid transformations that include rotation and translation. In general, a three dimensional Euclidean transformation is a function:

$$f : \mathbb{R}^3 \to \mathbb{R}^3. \tag{3.20}$$

Each Euclidean transformation can be decoupled into rotation and translation and can be parametrized by a minimum of six parameters – three for rotation and three for translation. Rotation parametrizations were discussed in the previous section, and for translation the most common representation is a three dimensional vector that specifies the displacement. It does not have discontinuities and uses exactly three parameters to specify three degrees of freedom.

Transformation Matrix is a 4×4 matrix that has the following structure

$$T = \begin{pmatrix} R \in \mathbf{SO}(3) & t \in \mathbb{R}^3 \\ 0^{1\times 3} & 1 \end{pmatrix},$$

with rotation matrix R and translation vector t. The coordinates (x', y', z') of the transformed point can be computed by pre-multiplying the homogeneous coordinates of the point (x, y, z) with the transformation matrix

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.21}$$

Several transformations can be concatenated by multiplying their transformation matrices and the inverse can be computed as follows

$$T^{-1} = \begin{pmatrix} R^T & -R^T t \\ 0^{1\times 3} & 1 \end{pmatrix}.$$

3.1.4 SE(3) Lie Group and $\mathfrak{se}(3)$ Lie Algebra

Transformation matrices form the **SE**(3) group under matrix multiplication and represent six degrees of freedom. Lie algebra elements can be parametrized with $\xi \in \mathbb{R}^6$ which can be transformed to an $\mathfrak{se}(3)$ element with the *hat* operator. Similar to **SO**(3) the mapping from $\mathfrak{se}(3)$ to **SE**(3) is called exponential map

$$T = e^{\hat{\xi}}, \tag{3.22}$$

$$\hat{\xi} = \sum_{i=1}^{6} G_i \xi(i), \tag{3.23}$$

where G_i is the i-th generator of the group

$$G_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.24}$$

$$G_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.25}$$

Again, the Rodrigues formula can be used to find a solution in closed form. The vector $\xi = (\omega \in \mathbb{R}^3, t' \in \mathbb{R}^3)$ consists of the translation and the rotation part, so the mapping to $\mathbf{SE}(3)$ can be done with the following equation:

$$T = e^{\hat{\xi}} = \begin{pmatrix} e^{\hat{\omega}} & Vt' \\ 0^{1\times 3} & 1 \end{pmatrix}, \tag{3.26}$$

where

$$V = \mathbf{I} + \frac{1-\cos(\theta)}{\theta^2}\hat{\omega} + \frac{\theta - \sin(\theta)}{\theta^3}\hat{\omega}^2, \tag{3.27}$$

$\theta = ||\omega||$, and $e^{\hat{\omega}}$ and $\hat{\omega}$ defined in Section 3.1.1. The log map also has a closed-form solution in $\mathbf{SE}(3)$ and is defined as

$$\xi = \log(T)^{\vee} = (\omega, t'), \tag{3.28}$$
$$\omega = \log(R)^{\vee}, \tag{3.29}$$
$$t' = V^{-1}t, \tag{3.30}$$

where log map for the rotation part is defined in Eq. 3.17.

The Jacobian of this function at $\xi = 0$ is given as follows

$$\left.\frac{\partial e^{\hat{\xi}}}{\partial \xi}\right|_{\xi=0} = \begin{pmatrix} | & | & | & | & | & | \\ G_1 & G_2 & G_3 & G_4 & G_5 & G_6 \\ | & | & | & | & | & | \end{pmatrix}. \tag{3.31}$$

3.2 Probability Theory and Optimization

3.2.1 Multivariate Gaussian

The common way to parameterize the multivariate Gaussian (normal) distribution density function is according to

$$x \sim N(x; \mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{3.32}$$

where x is a p dimensional vector that is Gaussian distributed, with mean μ and covariance Σ (symmetric positive-definite). This parametrization is called the *moment form*.

An alternative parametrization of the Gaussian density is called *information form* and is defined as follows

$$x \sim N^{-1}(x; v, \Lambda) = \frac{\exp\left(-\frac{1}{2}v^T\Lambda^{-1}v\right)}{(2\pi)^{p/2}|\Lambda|^{-1/2}} \exp\left(-\frac{1}{2}x^T\Lambda x + x^T v\right), \tag{3.33}$$

where v is called information vector and Λ is called information matrix.

It is easy to see that these two forms are equivalent with

$$v = \Sigma^{-1}\mu, \tag{3.34}$$
$$\Lambda = \Sigma^{-1}. \tag{3.35}$$

3.2.2 Maximum Likelihood Estimation

In this section we describe the maximum likelihood estimation (MLE) framework that is used throughout this thesis. The theory provided here can be found in probability theory and statistics textbooks, for example [106]. Some proofs are skipped for brevity, so we refer the reader to the textbooks.

Let $\mathbf{y} = (Y_1, ..., Y_n)$ be n independent random variables with probability density function defined as $f_i(y_i, \theta)$ that depends on the vector-valued parameter $\theta = (\theta_1, ..., \theta_p)$. The *likelihood* function is then defined as follows

$$L(\theta, \mathbf{y}) = \prod_{i=1}^{n} f_i(y_i, \theta). \tag{3.36}$$

We use the natural logarithm to turn the product into a sum and define the *log-likelihood* as

$$\log L(\theta, \mathbf{y}) = \sum_{i=1}^{n} \log f_i(y_i, \theta), \tag{3.37}$$

which has the same maximizer as the original likelihood function.

The maximum-likelihood estimate of the parameters $\hat{\theta}$ is the one that maximizes the likelihood. More formally

$$\log L(\hat{\theta}, \mathbf{y}) \geq \log L(\theta, \mathbf{y}), \forall \theta. \tag{3.38}$$

The vector of partial derivatives of the log-likelihood function $u(\theta)$ is called *score vector*. It has the size $p \times 1$ and is defined as

$$u(\theta) = \frac{\partial \log L(\theta, \mathbf{y})}{\partial \theta}, \tag{3.39}$$

and the *Fisher information matrix* is defined as a $p \times p$ matrix of second derivatives of the negative log-likelihood function:

$$I(\theta) = -\left\{ \frac{\partial^2 \log L(\theta, \mathbf{y})}{\partial \theta_i \partial \theta_j} \right\}. \tag{3.40}$$

If certain regularity conditions are met, the MLE estimate $\hat{\theta}$ is asymptotically ($n \to \infty$) normal with mean θ_t and variance-covariance matrix $I(\theta_t)^{-1}$ [62]. More formally

$$\hat{\theta} \sim N(\theta_t, I(\theta_t)^{-1}), \tag{3.41}$$

where N is a multivariate Gaussian distribution and θ_t is the true value of the parameter. In practice, since the true value of θ_t is not known, the distribution can be estimated by substituting the MLE value $\hat{\theta}$ to the information matrix $I(\hat{\theta})$. This matrix is called *observed sample Fisher information* and it asymptotically converges to the *Fisher information* matrix.

Note that until now we did not discuss how we can find the MLE solution. This will be discussed in Section 3.2.4 for the cases that are relevant for this thesis. An important result from this subsection is that the MLE solution is asymptotically normal and we can estimate the parameters of this normal distribution.

3.2.3 Partial Marginalization

After estimating the maximum likelihood solution $\hat{\theta}$ we obtain the estimated distribution of this solution that is a Gaussian with $\Sigma = I(\hat{\theta})^{-1}$. The probability density function is defined as follows

$$x \sim N(x; \hat{\theta}, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \hat{\theta})^T \Sigma^{-1} (x - \hat{\theta}) \right). \tag{3.42}$$

Splitting the parameter vector in two components $\hat{\theta} = [\hat{\theta}_a, \hat{\theta}_b]$ we are interested in the marginal distribution of $\hat{\theta}_a$. We can separate Σ into corresponding blocks

$$\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ab}^T & \Sigma_{bb} \end{bmatrix}, \tag{3.43}$$

and rewrite the probability distribution as

$$\begin{bmatrix} x_a \\ x_b \end{bmatrix} \sim \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_a - \hat{\theta}_a \\ x_b - \hat{\theta}_b \end{bmatrix}^T \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ab}^T & \Sigma_{bb} \end{bmatrix}^{-1} \begin{bmatrix} x_a - \hat{\theta}_a \\ x_b - \hat{\theta}_b \end{bmatrix} \right). \tag{3.44}$$

The marginal distribution of x_a can be found by selecting corresponding blocks of θ and Σ (for formal proof see [117])

$$x_a \sim N(x_a; \hat{\theta}_a, \Sigma_{aa}) = \frac{1}{(2\pi)^{p/2}|\Sigma_{aa}|^{1/2}} \exp\left(-\frac{1}{2}(x_a - \hat{\theta}_a)^T \Sigma_{aa}^{-1}(x_a - \hat{\theta}_a) \right). \tag{3.45}$$

If the distribution density function is provided in the *information form* $N^{-1}(x; v, \Lambda)$ (Equation 3.33), the marginal distribution is computed using the Schur complement as follows (for proof see [127])

$$x_a \sim N^{-1}(x_a; \bar{v}_a, \bar{\Lambda}_{aa}), \tag{3.46}$$

$$\bar{\Lambda}_{aa} = \Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ab}^T, \tag{3.47}$$

$$\bar{v}_a = v_a - \Lambda_{ab}\Lambda_{bb}^{-1}v_b, \tag{3.48}$$

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ab}^T & \Lambda_{bb} \end{bmatrix}, \tag{3.49}$$

$$v = \begin{bmatrix} v_a \\ v_b \end{bmatrix}. \tag{3.50}$$

3.2.4 Maximum Likelihood Estimate for Observations with Gaussian Noise

Until now we assumed we have a maximum-likelihood estimate for our problem, but we did not discuss how we can compute it. Assume we have a set of n independent observations $\mathbf{y} = (Y_1, ..., Y_n)$, where each observation has the form $Y_i = g_i(\theta) + N(0, \sigma_i^2)$. With Equations 3.38 and 3.32 we can write the log-likelihood as follows

$$\log L(\theta, \mathbf{y}) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2} \frac{(Y_i - g_i(\theta))^2}{\sigma_i^2} \right) \tag{3.51}$$

$$= C - \frac{1}{2} \sum_{i=1}^{n} \frac{(Y_i - g_i(\theta))^2}{\sigma_i^2}, \tag{3.52}$$

$$\tag{3.53}$$

where C is constant. Maximizing the log-likelihood is the same as minimizing the negative log-likelihood, so we can write

$$\hat{\theta} = \arg\min_{\theta} -\log L(\theta, \mathbf{y}) = \arg\min_{\theta} \frac{1}{2} \sum_{i=1}^{n} \frac{(Y_i - g_i(\theta))^2}{\sigma_i^2}, \tag{3.54}$$

where $\hat{\theta}$ is the maximum likelihood estimate of the parameter. This problem is known as a least-squares problem. It is a well studied problem for which efficient solving methods exist. They will be discussed in the following.

3.2.5 Gauss-Newton Method for Least-Squares Problems

Non-linear least-squares problems are widely used to fit the parameters of a function to measurement data. For a vector function $r : \mathbb{R}^p \to \mathbb{R}^m$ we are looking for the parameter vector θ that minimizes $||r(\theta)||^2$ [81], where the elements of the vector function $r_i(\theta) = \frac{Y_i - g_i(\theta)}{\sigma_i}$ are the residuals.

$$\hat{\theta} = \arg\min_{\theta} F(\theta), \tag{3.55}$$

where

$$F(\theta) = \frac{1}{2} \sum r_i(\theta)^2 = \frac{1}{2} r(\theta)^T r(\theta). \tag{3.56}$$

This is exacly the same minimization problem as in Eq. 3.54. The Gauss-Newton method uses a linear approximation of $r(\theta)$

$$r(\theta + h) \simeq l(h) = r(\theta) + J(\theta)h, \tag{3.57}$$

where J is the Jacobian of the vector function r, the entries in which can be computed as

$$J(\theta)_{ij} = \frac{\partial r_i}{\partial \theta_j}(\theta). \tag{3.58}$$

With this approximation

$$F(\theta + h) \simeq L(h) = \frac{1}{2} l(h)^T l(h) \tag{3.59}$$

$$= \frac{1}{2} r(\theta)^T r(\theta) + h^T J(\theta)^T r(\theta) + \frac{1}{2} h^T J(\theta)^T J(\theta) h \tag{3.60}$$

$$= F(\theta) + h^T J(\theta)^T r(\theta) + \frac{1}{2} h^T J(\theta)^T J(\theta) h. \tag{3.61}$$

$L(h)$ is a quadratic function function with derivative

$$L'(h) = J(\theta)^T r(\theta) + J(\theta)^T J(\theta) h. \tag{3.62}$$

We know that for the optimal value of h that minimizes $L(h)$ this derivative should be zero. According to that

$$J(\theta)^T J(\theta) h^* = -J(\theta)^T r(\theta) \tag{3.63}$$

As the function F is non-linear we have to iteratively linearize it around the current state and search for the optimal solution of the linear function.

$$\theta_{i+1} = \theta_i + h^* \tag{3.64}$$

If the curvature of the functions r_i is small this method shows quadratic convergence [81]. This usually holds when the state is close to the minimum of a smooth differentiable function. In this thesis the Gauss-Newton method will be used for the visual-inertial odometry algorithms described in the next chapters.

3.2.6 Gauss-Newton Method for Smooth Manifolds

For many problems in the real world the state variables that we want to estimate lie on a smooth manifold and not in Euclidean space. Some examples of such states include the rotations $\mathbf{SO}(3)$ and the rigid body transformations $\mathbf{SE}(3)$ as discussed in Section 3.1. Luckily the optimization method described in Section 3.2.5 can be easily adapted for such cases.

When optimizing on smooth manifolds we parametrize updates to the state in the tangent space and compute the Jacobians accordingly. We can then rewrite the linearization defined in Eq. 3.57 as

$$r(\theta \oplus h) \simeq l(h) = r(\theta) + J(\theta) h, \tag{3.65}$$

and rewrite the update defined in Eq. 3.64

$$\theta_{i+1} = \theta_i \oplus h^*. \tag{3.66}$$

One possible (but not unique) selection of the $\oplus$ operator is post-multiplication with the exponential map and the increment defined in Lie algebra tangent space. In the case of poses the state update can be represented as

$$\mathbf{T}_{i+1} = \mathbf{T}_i e^{\hat{h}^*}. \tag{3.67}$$

In this formulation the exponential map is linearized around zero at every iteration, so the Jacobians from Section 3.1 can be used for computing the Jacobians of the residual function using the chain rule.

3.2.7 Levenberg-Marquardt Method

Even though the Gauss-Newton algorithm described in Section 3.2.5 has quadratic convergence close to the optimum, for arbitrary initialization it can become unstable. It also does not guarantee that the value of the function that is being minimized will decrease at every iteration.

To overcome these limitations the Levenberg-Marquardt method can be used, which is a mixture of the gradient descent method that always decreases the function with sufficiently small step size, but has linear convergence, and the Gauss-Newton method with quadratic convergence. The update step is very similar to Eq. 3.63 and can be expressed as

$$h^* = -(J^T J + \lambda \operatorname{diag}(J^T J))^{-1} J^T r(\theta), \tag{3.68}$$

where diag denotes the diagonal matrix with the elements taken from the argument.

The algorithm to set the λ largely depends on the problem. One possible algorithm is described in [81] where the parameter is decreased if the linear approximation of the function (Eq. 3.57) behaves similar to the non-linear function, and increased if the behavior is different or the current step has increased the value of the minimized function.

3.2.8 Huber Norm and Iteratively Reweighted Least-Squares Problem

The methods described in the previous subsections minimize the sum of squared residuals and, because of that, are not robust to outliers. One way to make the methods more robust to outliers is to use a robust norm instead of the l_2 norm. In Chapters 5 and 6 we use the Huber norm to make the method robust to outliers. The Huber norm is defined as follows

$$\rho(r) = \begin{cases} \frac{1}{2}r^2, & \text{for } |r| \leq \sigma \\ \sigma|r| - \frac{1}{2}\sigma^2, & \text{for } |r| > \sigma \end{cases}, \tag{3.69}$$

where σ is the parameter of the Huber norm. For residuals smaller than σ it has the same properties as the l_2 norm, but for large residuals the loss becomes linear instead of quadratic. This allows to minimize the effect of outliers on the optimization procedure.

With that, we can formulate a robust optimization function as follows

$$\hat{\theta} = \arg\min_{\theta} F'(\theta), \tag{3.70}$$

$$F'(\theta) = \rho(r(\theta)), \tag{3.71}$$

where ρ is applied elementwise to all elements of the residual vector function $r(\theta)$. This function can be minimized using iteratively reweighted least-squares. We know

that the minimizer of the function should have the derivative equal to zero. We can write the derivative of the function F' using the chain rule as

$$\frac{\partial F'(\theta)}{\partial \theta} = \frac{\partial \rho(r)}{\partial r}\frac{\partial r(\theta)}{\partial \theta} = \frac{1}{r}\frac{\partial \rho(r)}{\partial r} r \frac{\partial r(\theta)}{\partial \theta} = \frac{1}{r}\frac{\partial \rho(r)}{\partial r}\frac{\partial F(\theta)}{\partial \theta} = 0, \tag{3.72}$$

where F is the original loss function with l_2 norm defined in 3.56. If we define the weight function as

$$w(r) = \frac{1}{r}\frac{\partial \rho(r)}{\partial r}, \tag{3.73}$$

we can see that equation 3.72 can be obtained by taking the derivative of the following function

$$F''(\theta) = \frac{1}{2}\sum w(r(\theta))r^2(\theta). \tag{3.74}$$

In iteratively reweighted least squares we fix $w(r(\theta))$ in every iteration and ignore its dependency on the residual for the Jacobian computation. This simplification turns the problem at every iteration into the simple weighted least-squares problem discussed in the previous subsections. Despite this approximation of the Jacobians, iteratively reweighted least squares shows fast convergence in practice.

Part II

Own Publications

Chapter 4

The Double Sphere Camera Model

Authors	Vladyslav Usenko[1]	usenko@in.tum.de
	Nikolaus Demmel [1]	demmeln@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1]Technische Universität München, Munich, Germany	

Publication: V. Usenko, N. Demmel and D. Cremers. "The Double Sphere Camera Model". In: *Proc. of the Int. Conference on 3D Vision (3DV)*. Sept. 2018. DOI: 10.1109/3DV.2018.00069. eprint: http://arxiv.org/abs/1807.08957

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 Vision-based motion estimation and 3D reconstruction, which have numerous applications (e.g., autonomous driving, navigation systems for airborne devices and augmented reality) are receiving significant research attention. To increase the accuracy and robustness, several researchers have recently demonstrated the benefit of using large field-of-view cameras for such applications.
In this paper, we provide an extensive review of existing models for large field-of-view cameras. For each model we provide projection and unprojection functions and the subspace of points that result in valid projection. Then, we propose the Double Sphere camera model that well fits with large field-of-view lenses, is computationally inexpensive and has a closed-form inverse. We evaluate the model using a calibration dataset with several different lenses and compare the models using the metrics that are relevant for Visual Odometry, i.e., reprojection error, as well as computation time for projection and unprojection functions and their Jacobians. We also provide qualitative results and discuss the performance of all models.

4.1 Introduction

Visual Odometry and Simultaneous Localization and Mapping are becoming important for numerous applications. To increase the accuracy and robustness of these methods, both hardware and software must be improved.

Several issues can be addressed by the use of large field-of-view cameras. First, with a large field-of-view, it is easier to capture more textured regions in the environment, which is required for stable vision-based motion estimation. Second, with a large field-of-view, large camera motions can be mapped to smaller pixel motions compared to cameras with a smaller field-of-view at the same resolution. This ensures small optical flow between consecutive frames, which is particularly beneficial for direct methods.

Previous studies have demonstrated that a large field-of-view is beneficial for vision-based motion estimation [132] [114]. Catadioptric cameras are mechanically complex and expensive; however fisheye lenses are small, lightweight, and widely available on the consumer market. Thus, in this paper we focus on fisheye lenses and models that describe their projection.

The reminder of this paper is organized as follows. In Section 8.2 we provide an extensive review of camera models that can be used with fisheye lenses. To make the paper self-contained we provide the projection and unprojection functions and define the subspace of valid projections for each model. In Section 4.3, we propose a novel projection model for fisheye cameras that has the following advantages.

- The proposed projection model is well suited to represent the distortions of fisheye lenses.

- The proposed model does not require computationally expensive trigonometric operations for projection and unprojection.
- Differing from projection models based on higher order polynomials [61] [116], that use iterative methods to unproject points, the inverse of the projection function exists in a closed form.

In Section 4.5, we evaluate all presented models with respect to metrics that are relevant for vision-based motion estimation. Here, we use a dataset collected using several different lenses to evaluate the reprojection error for each model. We also present the computation time required for projection and unprojection functions and the time required to compute Jacobians relative to their arguments.

The datasets used in this study together with the open-source implementation of the proposed model are available on the project page:

https://vision.in.tum.de/research/vslam/double-sphere

4.2 Related Work

We define the notations used in this paper prior to reviewing existing camera models that can be used with fisheye lenses. We use lowercase letters to denote scalars, e.g., u, bold uppercase letters to denote matrices, e.g., $\mathbf{R}$, and bold lowercase letters for vectors, e.g., $\mathbf{x}$.

Generally, we represent pixel coordinates as $\mathbf{u} = [u, v]^T \in \Theta \subset \mathbb{R}^2$, where Θ denotes the image domain to which points can be projected to. 3D point coordinates are denoted $\mathbf{x} = [x, y, z]^T \in \Omega \subset \mathbb{R}^3$, where Ω denotes the set of 3D points that result in valid projections.

For all camera models we assume all projections cross a single point (i.e., central projection) that defines the position of the camera coordinate frame. The orientation of the camera frame is defined as follows. The z axis is aligned with the principal axis of the camera, and two other orthogonal directions (x, y) align with the corresponding axes of the image plane. We define a coordinate frame rigidly attached to the calibration pattern such that the transformation $\mathbf{T}_{ca_n} \in SE(3)$, which is a matrix of the special Euclidean group, transforms a 3D coordinate from the calibration pattern coordinate system to the camera coordinate system for image n.

Generally, a camera projection function is a mapping $\pi : \Omega \to \Theta$. Its inverse $\pi^{-1} : \Theta \to \mathbb{S}^2$ unprojects image coordinates to the bearing vector of unit length, which defines a ray by which all points are projected to these image coordinates.

For all camera models discussed in this section, we provide definitions of π, π^{-1}, the vector of intrinsic parameters $\mathbf{i}$, Ω and Θ.

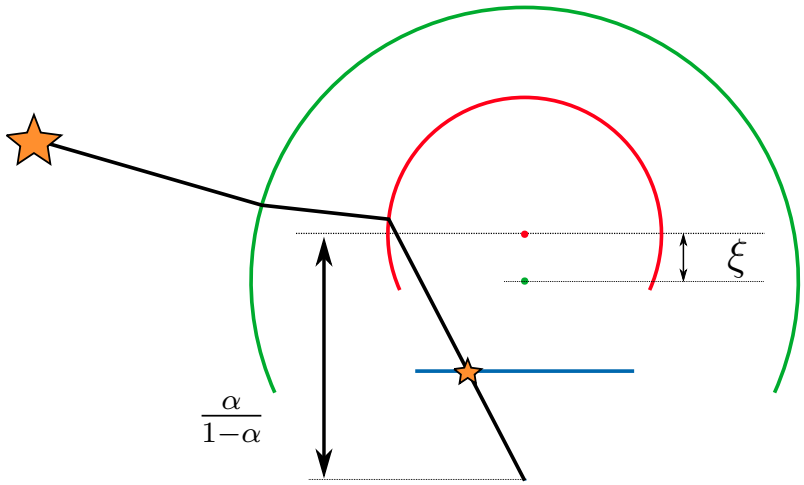
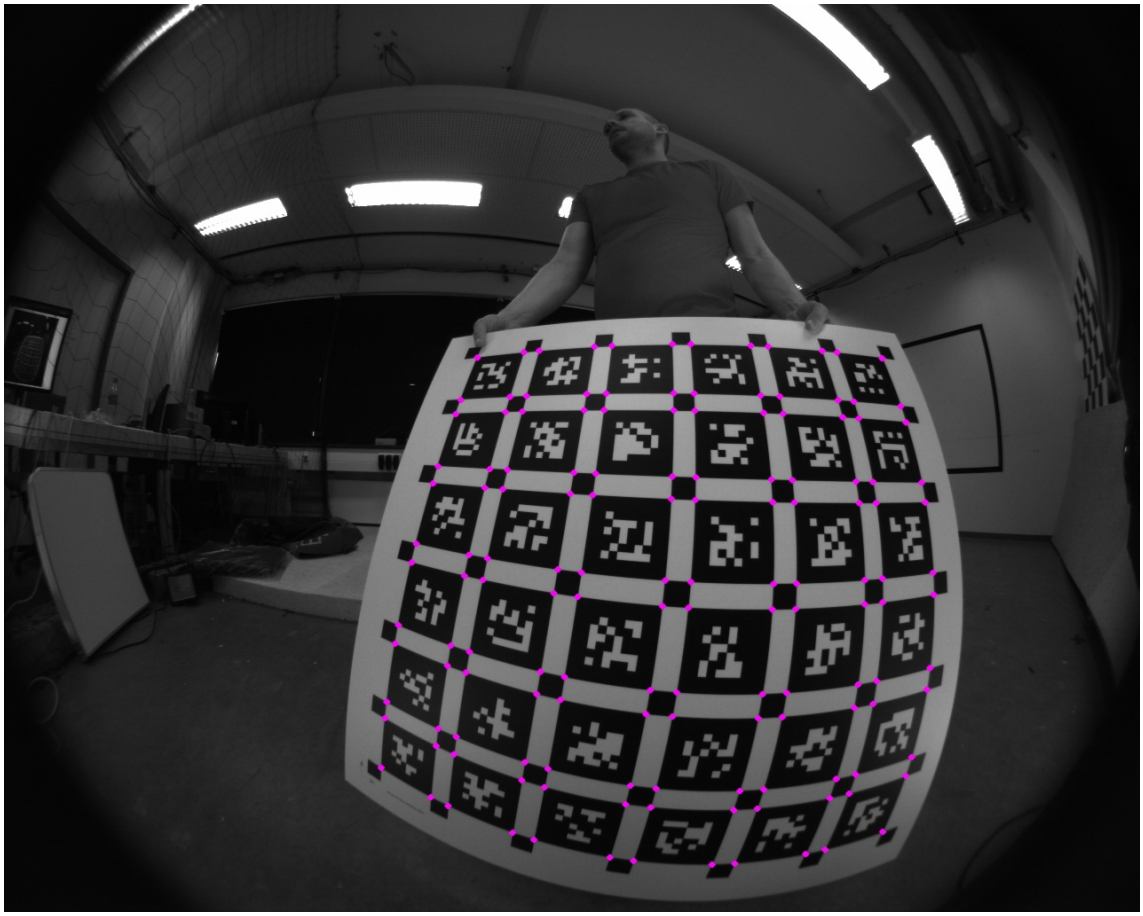


Figure 4.1: The proposed Double Sphere (**DS**) projection model. Initially, the point is projected onto the first sphere (green) and then onto the second sphere, which is shifted with respect to the first sphere by ξ (red). Then, the point is projected onto the image plane of a pinhole camera that is shifted by $\frac{\alpha}{1-\alpha}$ from the second sphere. The image below is the reprojection of the pattern corners after calibration using the proposed DS model, which indicates that the model fits the lens well.



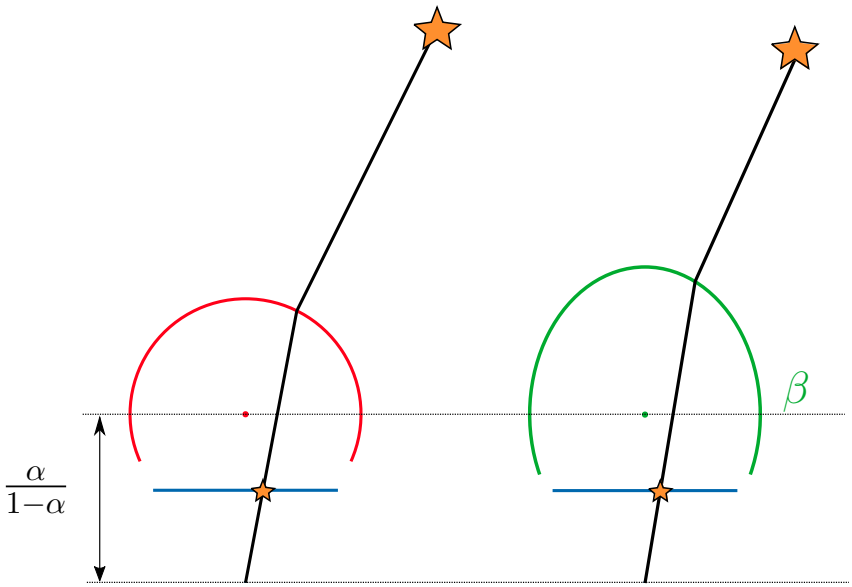


Figure 4.2: Schematic representation of the Unified Camera Model (**UCM**) and Extended Unified Camera Model (**EUCM**). First a 3D point is projected onto a unit sphere and then projected onto the image plane of the pinhole camera shifted by $\frac{\alpha}{1-\alpha}$ from the center of the sphere. In the EUCM, the sphere is transformed to an ellipsoid using the coefficient β.

4.2.1 Pinhole Camera Model

The pinhole camera model has four parameters $\mathbf{i} = [f_x, f_y, c_x, c_y]^T$ with a projection function that is defined as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.1}$$

It is easy to see that projection is defined for $\Omega = \{\mathbf{x} \in \mathbb{R}^3 \mid z > 0\}$, which theoretically limits the field-of-view to less than 180°. However, in practice, even when distortion model is added the pinhole camera demonstrates suboptimal performance for a field-of-view greater than 120°.

We can use the following function to unproject a point:

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \frac{1}{\sqrt{m_x^2 + m_y^2 + 1}} \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} \tag{4.2}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{4.3}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{4.4}$$

where unprojection is defined for $\Theta = \mathbb{R}^2$.

4.2.2 Unified Camera Model

The unified camera model (**UCM**) has five parameters $\mathbf{i} = [\gamma_x, \gamma_y, c_x, c_y, \xi]^T$ and is typically used with catadioptric cameras [86]. A previous study [47] has shown that the UCM can represent systems with parabolic, hyperbolic, elliptic and planar mirrors. This model can also be applied to cameras with fisheye lenses [131]. However, it does not fit most fisheye lenses perfectly; thus, an additional distortion model is often added.

In the UCM, projection is defined as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} \gamma_x \frac{x}{\xi d + z} \\ \gamma_y \frac{y}{\xi d + z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.5}$$

$$d = \sqrt{x^2 + y^2 + z^2}. \tag{4.6}$$

In this model, a point is first projected onto the unit sphere and then onto the image plane of the pinhole camera, which is shifted by ξ from the center of the unit sphere.

For practical applications we propose to rewrite this model as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{\alpha d + (1-\alpha) z} \\ f_y \frac{y}{\alpha d + (1-\alpha) z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{4.7}$$

This formulation of the model also has five parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, \alpha]^T$, $\alpha \in [0, 1]$ and is mathematically equivalent to the previous one ($\xi = \frac{\alpha}{1-\alpha}, \gamma_x = \frac{f_x}{1-\alpha}, \gamma_y = \frac{f_y}{1-\alpha}$). However, as discussed in Section 4.5, it has better numerical properties. Note that for $\alpha = 0$, the model degrades to the pinhole model.

The set of 3D points that result in valid projections is defined as follows:

$$\Omega = \{\mathbf{x} \in \mathbb{R}^3 \mid z > -wd\}, \tag{4.8}$$

$$w = \begin{cases} \frac{\alpha}{1-\alpha}, & \text{if } \alpha \leq 0.5, \\ \frac{1-\alpha}{\alpha} & \text{if } \alpha > 0.5, \end{cases} \tag{4.9}$$

where (for $\alpha > 0.5$) w represents the sine of the angle between the horizontal axis on schematic plot (Figure 4.2) and the perpendicular to the tangent of the circle from the focal point of the pinhole camera.

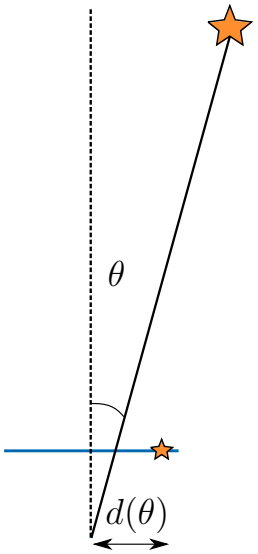


Figure 4.3: Schematic representation of the Kannala-Brandt Camera model (**KB**). The displacement of the projection from the optical center is proportional to $d(\theta)$, which is a polynomial function of the angle between the point and optical axis θ.

The unprojection function is defined as follows:

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \frac{\xi + \sqrt{1 + (1 - \xi^2)r^2}}{1 + r^2} \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix}, \tag{4.10}$$

$$m_x = \frac{u - c_x}{f_x}(1 - \alpha), \tag{4.11}$$

$$m_y = \frac{v - c_y}{f_y}(1 - \alpha), \tag{4.12}$$

$$r^2 = m_x^2 + m_y^2, \tag{4.13}$$

$$\xi = \frac{\alpha}{1 - \alpha}, \tag{4.14}$$

where Θ is defined as follows.

$$\Theta = \begin{cases} \mathbb{R}^2 & \text{if } \alpha \leq 0.5 \\ \{\mathbf{u} \in \mathbb{R}^2 \mid r^2 \leq \frac{(1-\alpha)^2}{2\alpha - 1}\} & \text{if } \alpha > 0.5 \end{cases} \tag{4.15}$$

4.2.3 Extended Unified Camera Model

A previous study [67] extended the unified camera model (**EUCM**) to have six parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, \alpha, \beta]^T$, $\alpha \in [0, 1]$, $\beta > 0$ and defined the following

projection function.

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{\alpha d + (1-\alpha) z} \\ f_y \frac{y}{\alpha d + (1-\alpha) z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.16}$$

$$d = \sqrt{\beta(x^2 + y^2) + z^2}. \tag{4.17}$$

The EUCM can be interpreted as a generalization of the UCM where the point is projected onto an ellipsoid symmetric around the z axis (Figure 4.2). That study also indicated that when treating the model as a projection on a quadratic surface followed by orthographic projection on the image plane the model is complete in the sense that it can represent all possible quadratic surfaces.

With EUCM, a set Ω is defined similar to the UCM, with the difference that d is computed by Eq. 4.17. Note that the EUCM degrades to a regular UCM for $\beta = 1$.

As mentioned previously, the EUCM projects on the ellipsoid. Therefore, the unit length vector for unprojection cannot be obtained directly; consequently, we must employ normalization. The unprojection function is defined as follows:

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \frac{1}{\sqrt{m_x^2 + m_y^2 + m_z^2}} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}, \tag{4.18}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{4.19}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{4.20}$$

$$r^2 = m_x^2 + m_y^2, \tag{4.21}$$

$$m_z = \frac{1 - \beta\alpha^2 r^2}{\alpha\sqrt{1 - (2\alpha - 1)\beta r^2} + (1 - \alpha)}, \tag{4.22}$$

where Θ is defined as follows.

$$\Theta = \begin{cases} \mathbb{R}^2 & \text{if } \alpha \leq 0.5 \\ \{\mathbf{u} \in \mathbb{R}^2 \mid r^2 \leq \frac{1}{\beta(2\alpha - 1)}\} & \text{if } \alpha > 0.5 \end{cases} \tag{4.23}$$

4.2.4 Kannala-Brandt Camera Model

The previous study [61] proposed the Kannala-Brandt (**KB**) model, which is a generic camera model that well fits regular, wide angle and fisheye lenses. The KB model assumes that the distance from the optical center of the image to the projected point is proportional to the polynomial of the angle between the point and the principal axis (Figure 4.3). We evaluate two versions of the KB

model, i.e.,: with six parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, k_1, k_2]^T$ and eight parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]^T$. The projection function of the KB model is defined as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x\ d(\theta)\ \frac{x}{r} \\ f_y\ d(\theta)\ \frac{y}{r} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.24}$$

$$r = \sqrt{x^2 + y^2}, \tag{4.25}$$

$$\theta = \text{atan2}(r, z), \tag{4.26}$$

$$d(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9, \tag{4.27}$$

assuming that polynomial $d(\theta)$ is monotonic $\Omega = \mathbb{R}^3 \setminus [0, 0, 0]^T$.

The unprojection function of the KB model requires finding the root of a high-order polynomial to recover angle θ from $d(\theta)$. This can be achieved through an iterative optimization, e.g., Newton's method. The unprojection function can be expressed as follows:

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \begin{bmatrix} \sin(\theta^*)\ \frac{m_x}{r_u} \\ \sin(\theta^*)\ \frac{m_y}{r_u} \\ \cos(\theta^*) \end{bmatrix}, \tag{4.28}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{4.29}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{4.30}$$

$$r_u = \sqrt{m_x^2 + m_y^2}, \tag{4.31}$$

$$\theta^* = d^{-1}(r_u), \tag{4.32}$$

where θ^* is the solution of $d(\theta) = r_u$. If $d(\theta)$ is monotonic $\Theta = \mathbb{R}^2$.

The KB model is sometimes used as a distortion model for a pinhole camera, e.g., the *equidistant* distortion model in Kalibr[1] [43] or the *fisheye* camera model in OpenCV[2]. The model is mathematically the same; however, since it first projects the point using the pinhole model and then applies distortion, it has a singularity at $z = 0$, which makes it unsuitable for fisheye lenses with field-of-view close to 180°when implemented is this manner.

Several other models for large field-of-view lenses based on high-order polynomials exist. For example, the main differences between [116] and the KB model are as follows: the model calibrates two separate polynomials for projection and unprojection to provide a closed-form solution for both, and for projection it uses the angle between the image plane and the point rather than of the angle between

[1] `https://github.com/ethz-asl/kalibr`

[2] `https://github.com/opencv/opencv`

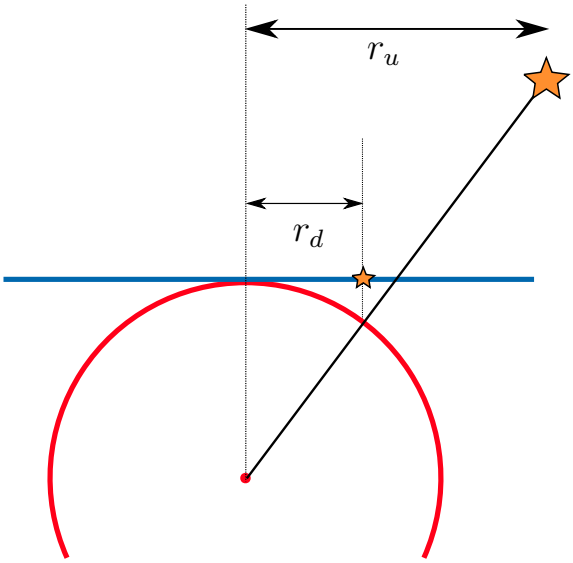


Figure 4.4: Schematic representation of the Field-of-View Camera model (**FOV**). Displacement of the projection from the optical center is proportional to the angle between the point and optical axis

the optical axis and the point. We expect this model to have similar performance and do not explicitly include it in our evaluation.

4.2.5 Field-of-View Camera Model

A previously proposed Field-of-view camera model (**FOV**) [30], has five parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, w]^T$ and assumes the distance between an image point and the principal point is typically approximately proportional to the angle between the corresponding 3D point and the optical axis (Figure 4.4). According to authors, parameter w approximately corresponds to the field-of-view of an ideal fisheye lens. The projection function for this model is defined as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \, r_d \, \frac{x}{r_u} \\ f_y \, r_d \, \frac{y}{r_u} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.33}$$

$$r_u = \sqrt{x^2 + y^2}, \tag{4.34}$$

$$r_d = \frac{\operatorname{atan2}(2 r_u \tan \frac{w}{2}, z)}{w}, \tag{4.35}$$

where $\Omega = \mathbb{R}^3 \setminus [0, 0, 0]^T$.

The FOV model has a closed-form solution for unprojecting the points, which is defined as follows:

Figure 4.5: Lenses used to evaluate camera models; left to right: BF2M2020S23 (195°), BF5M13720 (183°), BM4018S118 (126°), BM2820 (122°), and GoPro replacement lens (150°).

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \begin{bmatrix} m_x \frac{\sin(r_d w)}{2 r_d \tan\frac{w}{2}} \\ m_y \frac{\sin(r_d w)}{2 r_d \tan\frac{w}{2}} \\ \cos(r_d w) \end{bmatrix}, \tag{4.36}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{4.37}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{4.38}$$

$$r_d = \sqrt{m_x^2 + m_y^2}, \tag{4.39}$$

where $\Theta = \mathbb{R}^2$.

Similar to the KB model, the FOV model can be used as a distortion model for a pinhole camera.

4.3 Double Sphere Camera Model

We propose the Double Sphere (**DS**) camera model that better fits cameras with fisheye lenses, has a closed-form inverse, and does not require computationally expensive trigonometric operations. In the proposed DS model a point is consecutively projected onto two unit spheres with centers shifted by ξ. Then, the point is projected onto the image plane using the pinhole model shifted by $\frac{\alpha}{1-\alpha}$ (Figure 4.1). This model has six parameters $\mathbf{i} = [f_x, f_y, c_x, c_y, \xi, \alpha]^T$ and a projection function defined as follows:

Dataset	UCM [86] 5 parameters	FOV [30] 5 parameters	DS (Ours) 6 parameters	EUCM [67] 6 parameters	KB [61] 6 parameters	KB [61] 8 parameters
BF2M2020S23-1	0.236 (63.13%)	0.417 (187.90%)	0.145 (0.35%)	0.145 (0.30%)	0.164 (13.53%)	0.145 (0.00%)
BF2M2020S23-2	0.250 (59.94%)	0.490 (213.34%)	0.157 (0.23%)	0.157 (0.49%)	0.180 (15.43%)	0.156 (0.00%)
BF2M2020S23-3	0.277 (53.99%)	0.454 (151.81%)	0.180 (0.11%)	0.181 (0.47%)	0.202 (11.91%)	0.180 (0.00%)
BF5M13720-1	0.228 (49.53%)	0.307 (101.14%)	0.153 (0.00%)	0.154 (0.51%)	0.161 (5.41%)	0.153 (0.03%)
BF5M13720-2	0.250 (48.68%)	0.379 (124.91%)	0.169 (0.64%)	0.171 (1.73%)	0.183 (8.39%)	0.168 (0.00%)
BF5M13720-3	0.252 (54.99%)	0.386 (137.56%)	0.163 (0.56%)	0.165 (1.64%)	0.176 (8.51%)	0.162 (0.00%)
BM2820-1	0.238 (50.35%)	0.193 (22.10%)	0.159 (0.37%)	0.159 (0.34%)	0.159 (0.52%)	0.158 (0.00%)
BM2820-2	0.201 (60.13%)	0.163 (29.80%)	0.127 (0.90%)	0.127 (0.55%)	0.127 (0.54%)	0.126 (0.00%)
BM2820-3	0.227 (47.98%)	0.186 (21.13%)	0.154 (0.16%)	0.154 (0.15%)	0.154 (0.31%)	0.153 (0.00%)
BM4018S118-1	0.211 (11.76%)	0.208 (10.18%)	0.189 (0.03%)	0.189 (0.08%)	0.189 (0.15%)	0.189 (0.00%)
BM4018S118-2	0.247 (8.79%)	0.245 (8.19%)	0.227 (0.04%)	0.227 (0.02%)	0.227 (0.03%)	0.227 (0.00%)
BM4018S118-3	0.207 (13.69%)	0.205 (12.41%)	0.182 (0.02%)	0.182 (0.08%)	0.183 (0.17%)	0.182 (0.00%)
GOPRO-1	0.201 (36.84%)	0.150 (2.17%)	0.147 (0.04%)	0.147 (0.06%)	0.147 (0.30%)	0.147 (0.00%)
GOPRO-2	0.165 (30.52%)	0.128 (1.32%)	0.127 (0.00%)	0.127 (0.02%)	0.127 (0.25%)	0.127 (0.13%)
GOPRO-3	0.235 (40.41%)	0.171 (2.17%)	0.167 (0.09%)	0.168 (0.41%)	0.169 (1.02%)	0.167 (0.00%)
EUROC	0.137 (4.64%)	0.133 (1.21%)	0.131 (0.19%)	0.131 (0.25%)	0.132 (0.31%)	0.131 (0.00%)

Table 4.1: Mean reprojection error for evaluated camera models (in pixels). Best and second-best results are shown in green and orange, respectively. The table also shows overhead in % compared to the model with the smallest reprojection error in the sequence. The results show that the proposed model, despite having only six parameters, has less than 1% greater mean reprojection error than the best performing model with eight parameters.

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \\ f_y \frac{y}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.40}$$

$$d_1 = \sqrt{x^2 + y^2 + z^2}, \tag{4.41}$$

$$d_2 = \sqrt{x^2 + y^2 + (\xi d_1 + z)^2}. \tag{4.42}$$

A set of 3D points that results in valid projection is expressed as follows:

$$\Omega = \{\mathbf{x} \in \mathbb{R}^3 \mid z > -w_2 d_1\} \tag{4.43}$$

$$w_2 = \frac{w_1 + \xi}{\sqrt{2w_1\xi + \xi^2 + 1}} \tag{4.44}$$

$$w_1 = \begin{cases} \frac{\alpha}{1-\alpha}, & \text{if } \alpha \leq 0.5 \\ \frac{1-\alpha}{\alpha} & \text{if } \alpha > 0.5 \end{cases} \tag{4.45}$$

The unprojection function is computed as follows:

$$\pi^{-1}(\mathbf{u}, \mathbf{i}) = \frac{m_z\xi + \sqrt{m_z^2 + (1-\xi^2)r^2}}{m_z^2 + r^2} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix}, \tag{4.46}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{4.47}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{4.48}$$

$$r^2 = m_x^2 + m_y^2, \tag{4.49}$$

$$m_z = \frac{1 - \alpha^2 r^2}{\alpha\sqrt{1 - (2\alpha - 1)r^2} + 1 - \alpha}, \tag{4.50}$$

where the following holds.

$$\Theta = \begin{cases} \mathbb{R}^2 & \text{if } \alpha \leq 0.5 \\ \{\mathbf{u} \in \mathbb{R}^2 \mid r^2 \leq \frac{1}{2\alpha - 1}\} & \text{if } \alpha > 0.5 \end{cases} \tag{4.51}$$

4.4 Calibration

To estimate the camera parameters of each model we use a grid of AprilTag markers [104] (Figure 4.1) that can be detected automatically in the images. For each image n in the calibration sequence, the detection gives us the 2D position $\mathbf{u}_{nk}$ of the projection of corner k onto the image plane and the associated 3D location $\mathbf{x}_k$ of

the corner. After initial marker detection we use local subpixel refinement for each corner to achieve better calibration accuracy.

We formulate the optimization function that depends on the state $\mathbf{s} = [\mathbf{i}, \mathbf{T}_{ca_1}, ..., \mathbf{T}_{ca_N}]$ as follows:

$$E(s) = \sum_{n=1}^{N} \sum_{k \in K} \rho \left((\pi(\mathbf{T}_{ca_n} \mathbf{x}_k, \mathbf{i}) - \mathbf{u}_{nk})^2 \right), \tag{4.52}$$

where $\mathbf{i}$ is the vector of intrinsic parameters, π is the projection function, $\mathbf{T}_{ca_n} \in SE(3)$ is the transformation from the coordinate frame of the calibration grid to the camera coordinate frame for image n. K is a set of detected corner points for the image n and ρ is the robust Huber norm.

We parameterize the updates to the state with vector $\Delta\mathbf{s} = [\Delta\mathbf{i}, \Delta\mathbf{t}_0, ..., \mathbf{t}_N]^T$ as follows:

$$\mathbf{s} \oplus \Delta\mathbf{s} = \begin{bmatrix} \mathbf{i} + \Delta\mathbf{i} \\ \mathbf{T}_{ca_1} \exp(\Delta\mathbf{t}_1) \\ ... \\ \mathbf{T}_{ca_N} \exp(\Delta\mathbf{t}_N) \end{bmatrix} \tag{4.53}$$

Given the current state $\mathbf{s}_l$ we can rewrite the optimization function as:

$$E(\mathbf{s}_l \oplus \Delta\mathbf{s}) = \mathbf{r}(\mathbf{s}_l \oplus \Delta\mathbf{s})^T \mathbf{W} \mathbf{r}(\mathbf{s}_l \oplus \Delta\mathbf{s}), \tag{4.54}$$

and use the Gauss-Newton algorithm to compute the update for the current iteration as follows:

$$\Delta\mathbf{s} = (\mathbf{J}_l^T \mathbf{W} \mathbf{J}_l)^{-1} \mathbf{J}_l^T \mathbf{W} \mathbf{r}_l, \tag{4.55}$$

where $\mathbf{r}_l$ is a stacked vector of residuals evaluated at $\mathbf{s}_l$, $\mathbf{J}_l$ is the Jacobian of residuals with respect to $\Delta\mathbf{s}$, and $\mathbf{W}$ is the weighting matrix corresponding to the Huber norm. With that, we update the current estimate of the state

$$\mathbf{s}_{l+1} = \mathbf{s}_l \oplus \Delta\mathbf{s}, \tag{4.56}$$

and iterate until convergence.

Since the optimization function is non-convex, good initialization of the intrinsic parameters $\mathbf{i}$ and camera poses $\mathbf{T}_{ca}$ is important for optimization to converge. We initialize the intrinsic parameters with using the previously proposed method [51] (with $\beta = 1$ for EUCM and $\xi = 0$ for DS) and find initial poses using the UPnP algorithm [69].

Expressions Computed	UCM	FOV	DS	EUCM	KB 6	KB 8
$\pi(\mathbf{x}, \mathbf{i})$	33.842	419.339	55.020	32.965	288.003	305.841
$\pi(\mathbf{x}, \mathbf{i})$, $\mathbf{J_x}$, $\mathbf{J_i}$	34.555	433.956	55.673	33.534	293.625	310.399
$\pi^{-1}(\mathbf{u}, \mathbf{i})$	71.945	430.109	107.054	92.735	561.174	638.150
$\pi^{-1}(\mathbf{u}, \mathbf{i})$, $\mathbf{J_u}$, $\mathbf{J_i}$	71.079	891.556	181.119	95.883	537.291	613.287

Table 4.2: Timing for 10000 operations in microseconds measured on Intel Xeon E5-1620. **J** denotes the Jacobian of the function. The results demonstrate that with similar accuracy, our model shows around five times faster computation time for the projection function than the KB 8 model.

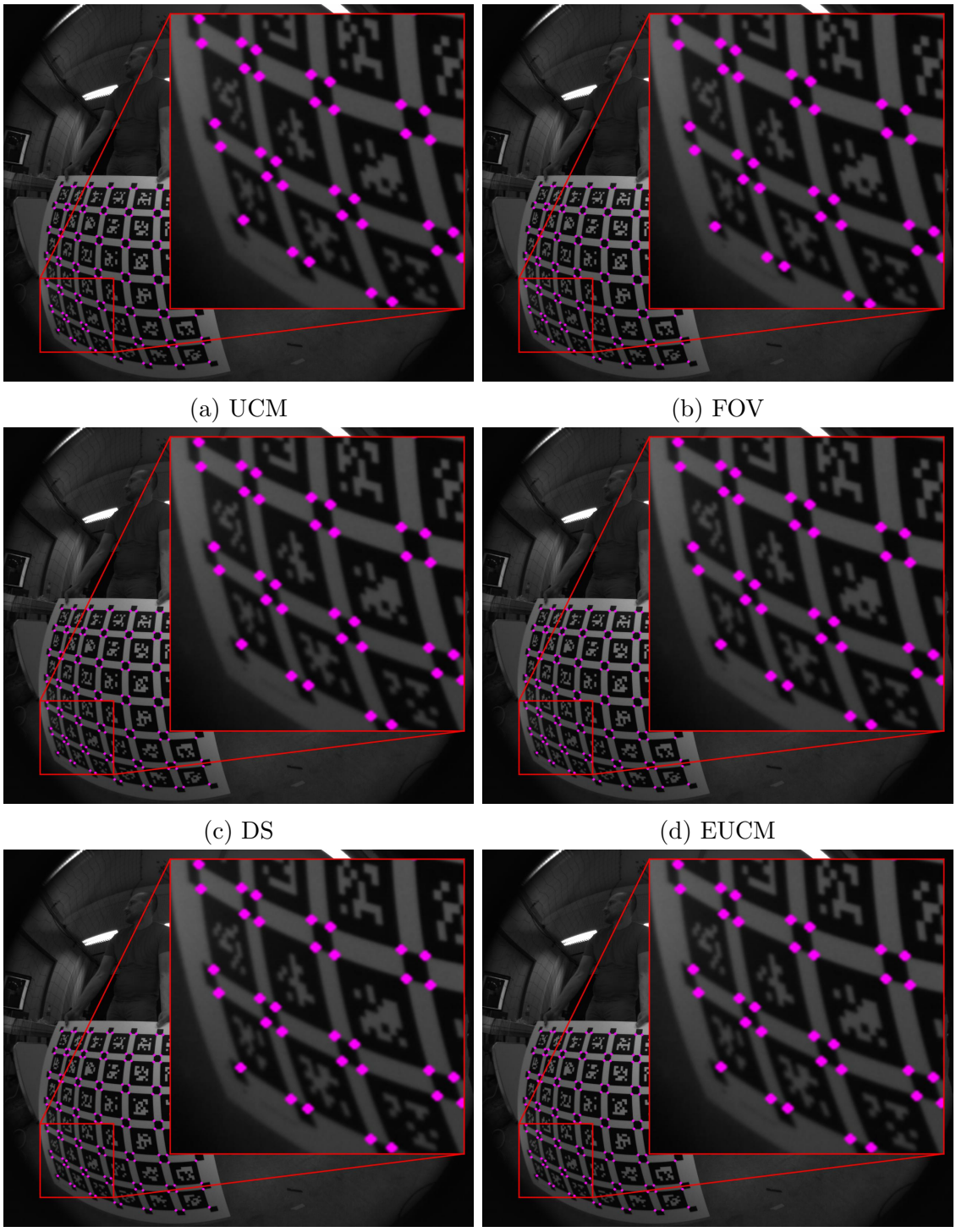
4.5 Evaluation

We evaluate the presented camera models using a dataset with 16 sequences. This dataset contains calibration sequences captured with five different lenses (three sequences for each lens) and one calibration sequence from the EuRoC dataset [20]. The lenses used to collect the sequences are shown in Figure 4.5. To ensure fair comparison, we first detect the calibration corners from all sequences and perform the optimization described in Section 4.4 using the same data for all models.

Reprojection error which indicates how well a model can represent the projection function of the actual lens, is one of the most important metrics for a camera model. Table 4.1 shows the mean reprojection error after optimizing for poses and intrinsic parameters computed for all datasets using different camera models. The best and second-best results for each sequence are shown in green and orange, respectively. For all entries, we also provide overhead computed as $\frac{c-b}{b} \times 100\%$, where b is the smallest reprojection error in the sequence and c is the reprojection error of the current model.

With most of the sequences, the KB model with eight parameters shows the best result, and the proposed model (DS) is the second best. Despite the fact the KB model has eight intrinsic parameters compared to six in the proposed DS model, the reprojection error overhead is less than 1% for all sequences. The EUCM demonstrates slightly greater reprojection error than that of the DS model and smaller reprojection error than the KB model with six parameters. The UCM and FOV models show greater reprojection errors among all tested models.

Computation time is another important aspect of a camera model because projection and unprojection functions are evaluated thousands of times in each iteration of vision-based motion estimation. Moreover, for optimization algorithms we must compute the Jacobians of these functions relative to the points and intrinsic param-



(a) UCM (b) FOV

(c) DS (d) EUCM

(e) KB 6 (f) KB 8

Figure 4.6: Corners of the calibration pattern (purple) projected onto the image after optimizing camera poses and intrinsic parameters for different camera models. The DS, EUCM and KB 8 models show high reprojection accuracy, while the UCM and KB 6 models have slightly shifted corner positions at the bottom-left corner of the calibration pattern. For the FOV model, displacement of the bottom-left corner is clearly visible, which indicates this model does not well fit the lens.

eters; thus, the computation time of these operations should also be considered.

Table 4.2 summarizes the computation times of those operations for the presented models. For each camera model, we measure the time of 10000 operations using the Google Benchmark[3] library on an Intel Xeon E5-1620 CPU. To compile the benchmarks, we use GCC 7 with O3 optimization level and execute the code in a single thread. Note that a small time difference between computing only the function and computing the function with Jacobians can be explained by the superscalar architecture of modern CPUs, which parallelizes execution internally.

The timing results show that the FOV and KB models are much slower than the other models. For example, the KB model with eight parameters is approximately nine times slower than the EUCM and five times slower than the DS model when evaluating the projection function. This is due to the fact that the KB model involves computationally expensive trigonometric operations ($atan2$).

Unprojection in KB models require iterative optimization to solve the polynomial roots, which together with the trigonometric operations, makes it several times slower than the UCM, EUCM and DS models. The FOV model is the slowest relative to unprojection, which is likely due to its multiple trigonometric operations.

Qualitative results of reprojection quality for the evaluated models are shown in Figure 4.6. Here, we project the corners of the calibration pattern after optimizing for pose and intrinsic parameters and visualize them on the corresponding image taken from the BF2M2020S23-3 sequence. The DS EUCM and KB 8 models provide similar results that are difficult to distinguish by the human eye. The UCM and KB 6 model well fit the corners in the middle of the image; however, these models have a small shift close to the edges. Note that imperfections are clearly visible with the FOV model.

Different formulations of UCM are evaluated in terms of the numerical stability of the results. Table 4.3 shows the mean and standard deviation (in %) of the intrinsic parameters computed on three different sequences for each lens. For the UCM we provide two formulations with the same reprojection error that are formulated with different intrinsic parameters. The results for the standard formulation as defined in the literature [86] ($\mathbf{i} = [\gamma_x, \gamma_y, c_x, c_y, \xi]^T$) are presented in the second column and show higher standard deviation than the results of the model parametrized with $\mathbf{i} = [f_x, f_y, c_x, c_y, \alpha]^T$. This can be explained by the strong coupling between γ_x, γ_y and ξ, which is not the case for the proposed parametrization. Moreover, for this formulation the focal length stays close to the focal length of the other camera models.

[3] `https://github.com/google/benchmark`

4.6 Conclusion

In this paper, we present the novel Double Sphere camera model that is well suited to fisheye cameras. We compare the proposed camera model to other state-of-the-art camera models. In addition, we provide an extensive evaluation of the presented camera models using 16 different calibration sequences and six different lenses. The evaluation results demonstrate that the model based on high-order polynomials (i.e., KB 8) shows the lowest reprojection error but is 5-10 times slower than competing models. Both the proposed DS model and the EUCM show very low reprojection error, with the DS model being slightly more accurate (less than 1% greater reprojection error compared to KB 8 on all sequences), and the EUCM being slightly faster (nine times faster projection evaluation than KB 8). Moreover, both models have a closed-form inverse and do not require computationally expensive trigonometric operations.

These results demonstrate that models based on spherical projection present a good alternative to models based on high-order polynomials for applications where fast projection, unprojection and a closed-form inverse are required.

Lens	**UCM** $[f_x, f_y, c_x, c_y, \alpha]^T$	**UCM** $[\gamma_x, \gamma_y, c_x, c_y, \xi]^T$	**FOV** $[f_x, f_y, c_x, c_y, w]^T$	**DS** $[f_x, f_y, c_x, c_y, \xi, \alpha]^T$	**EUCM** $[f_x, f_y, c_x, c_y, \alpha, \beta]^T$	**KB 6** $[f_x, f_y, c_x, c_y, k_1, k_2]^T$	**KB 8** $[f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]^T$
BF2M2020S23	$\begin{pmatrix}377.60\\377.48\\638.74\\514.00\\0.64\end{pmatrix} \pm \begin{pmatrix}0.20\%\\0.23\%\\0.05\%\\0.04\%\\0.16\%\end{pmatrix}$	$\begin{pmatrix}1041.97\\1041.63\\638.74\\514.00\\1.76\end{pmatrix} \pm \begin{pmatrix}0.48\%\\0.50\%\\0.05\%\\0.04\%\\0.44\%\end{pmatrix}$	$\begin{pmatrix}352.58\\352.72\\638.23\\513.08\\0.93\end{pmatrix} \pm \begin{pmatrix}0.17\%\\0.16\%\\0.08\%\\0.20\%\\0.02\%\end{pmatrix}$	$\begin{pmatrix}313.21\\313.21\\638.66\\514.39\\-0.18\\0.59\end{pmatrix} \pm \begin{pmatrix}0.11\%\\0.11\%\\0.01\%\\0.03\%\\0.68\%\\0.05\%\end{pmatrix}$	$\begin{pmatrix}380.95\\380.94\\638.66\\514.37\\0.63\\1.04\end{pmatrix} \pm \begin{pmatrix}0.04\%\\0.04\%\\0.01\%\\0.03\%\\0.03\%\\0.06\%\end{pmatrix}$	$\begin{pmatrix}380.14\\380.10\\638.65\\514.30\\0.01\\-0.01\end{pmatrix} \pm \begin{pmatrix}0.01\%\\0.00\%\\0.01\%\\0.02\%\\4.04\%\\3.51\%\end{pmatrix}$	$\begin{pmatrix}380.99\\380.98\\638.66\\514.38\\0.01\\-0.00\\0.00\\-0.00\end{pmatrix} \pm \begin{pmatrix}0.02\%\\0.03\%\\0.01\%\\0.03\%\\6.35\%\\22.04\%\\511.08\%\\15.18\%\end{pmatrix}$
BM2820	$\begin{pmatrix}528.31\\528.46\\624.08\\512.58\\0.64\end{pmatrix} \pm \begin{pmatrix}0.20\%\\0.19\%\\0.05\%\\0.03\%\\0.33\%\end{pmatrix}$	$\begin{pmatrix}1470.51\\1470.93\\624.08\\512.58\\1.78\end{pmatrix} \pm \begin{pmatrix}0.78\%\\0.77\%\\0.05\%\\0.03\%\\0.92\%\end{pmatrix}$	$\begin{pmatrix}491.60\\491.71\\624.20\\512.68\\0.92\end{pmatrix} \pm \begin{pmatrix}0.14\%\\0.14\%\\0.05\%\\0.04\%\\0.14\%\end{pmatrix}$	$\begin{pmatrix}386.17\\386.23\\624.29\\512.49\\-0.27\\0.55\end{pmatrix} \pm \begin{pmatrix}0.20\%\\0.21\%\\0.04\%\\0.02\%\\0.38\%\\0.12\%\end{pmatrix}$	$\begin{pmatrix}530.18\\530.27\\624.28\\512.49\\0.57\\1.17\end{pmatrix} \pm \begin{pmatrix}0.08\%\\0.09\%\\0.04\%\\0.02\%\\0.14\%\\0.06\%\end{pmatrix}$	$\begin{pmatrix}530.09\\530.18\\624.28\\512.49\\-0.00\\0.01\end{pmatrix} \pm \begin{pmatrix}0.09\%\\0.09\%\\0.04\%\\0.02\%\\140.31\%\\0.26\%\end{pmatrix}$	$\begin{pmatrix}530.35\\530.44\\624.29\\512.48\\-0.01\\0.02\\-0.02\\0.01\end{pmatrix} \pm \begin{pmatrix}0.08\%\\0.09\%\\0.04\%\\0.02\%\\15.18\%\\14.05\%\\20.23\%\\21.39\%\end{pmatrix}$
BF5M13720	$\begin{pmatrix}258.53\\258.45\\637.53\\511.89\\0.65\end{pmatrix} \pm \begin{pmatrix}0.07\%\\0.05\%\\0.01\%\\0.06\%\\0.03\%\end{pmatrix}$	$\begin{pmatrix}741.24\\741.03\\637.53\\511.89\\1.87\end{pmatrix} \pm \begin{pmatrix}0.10\%\\0.09\%\\0.01\%\\0.06\%\\0.07\%\end{pmatrix}$	$\begin{pmatrix}242.16\\242.18\\637.51\\512.21\\0.95\end{pmatrix} \pm \begin{pmatrix}0.09\%\\0.11\%\\0.03\%\\0.05\%\\0.15\%\end{pmatrix}$	$\begin{pmatrix}208.36\\208.35\\637.45\\512.18\\-0.20\\0.59\end{pmatrix} \pm \begin{pmatrix}0.14\%\\0.14\%\\0.01\%\\0.02\%\\0.41\%\\0.06\%\end{pmatrix}$	$\begin{pmatrix}260.67\\260.66\\637.45\\512.17\\0.64\\1.06\end{pmatrix} \pm \begin{pmatrix}0.07\%\\0.07\%\\0.01\%\\0.02\%\\0.09\%\\0.11\%\end{pmatrix}$	$\begin{pmatrix}260.28\\260.27\\637.45\\512.15\\0.00\\-0.00\end{pmatrix} \pm \begin{pmatrix}0.06\%\\0.06\%\\0.01\%\\0.02\%\\23.96\%\\3.31\%\end{pmatrix}$	$\begin{pmatrix}260.87\\260.86\\637.45\\512.19\\-0.01\\-0.00\\-0.00\\-0.00\end{pmatrix} \pm \begin{pmatrix}0.10\%\\0.10\%\\0.01\%\\0.02\%\\33.44\%\\584.87\%\\741.92\%\\65.29\%\end{pmatrix}$
GOPRO	$\begin{pmatrix}499.67\\499.78\\620.72\\513.74\\0.68\end{pmatrix} \pm \begin{pmatrix}0.03\%\\0.04\%\\0.07\%\\0.16\%\\0.13\%\end{pmatrix}$	$\begin{pmatrix}1546.22\\1546.54\\620.72\\513.74\\2.09\end{pmatrix} \pm \begin{pmatrix}0.30\%\\0.31\%\\0.07\%\\0.16\%\\0.41\%\end{pmatrix}$	$\begin{pmatrix}462.90\\462.94\\621.07\\513.36\\0.95\end{pmatrix} \pm \begin{pmatrix}0.02\%\\0.03\%\\0.06\%\\0.10\%\\0.00\%\end{pmatrix}$	$\begin{pmatrix}364.84\\364.86\\621.12\\513.27\\-0.27\\0.57\end{pmatrix} \pm \begin{pmatrix}0.09\%\\0.08\%\\0.06\%\\0.10\%\\0.26\%\\0.06\%\end{pmatrix}$	$\begin{pmatrix}501.02\\501.06\\621.12\\513.26\\0.60\\1.17\end{pmatrix} \pm \begin{pmatrix}0.03\%\\0.04\%\\0.06\%\\0.10\%\\0.22\%\\0.26\%\end{pmatrix}$	$\begin{pmatrix}500.92\\500.96\\621.10\\513.26\\-0.02\\0.00\end{pmatrix} \pm \begin{pmatrix}0.03\%\\0.04\%\\0.06\%\\0.10\%\\1.65\%\\6.96\%\end{pmatrix}$	$\begin{pmatrix}501.13\\501.17\\621.12\\513.27\\-0.02\\0.01\\-0.00\\0.00\end{pmatrix} \pm \begin{pmatrix}0.04\%\\0.05\%\\0.06\%\\0.09\%\\6.47\%\\41.92\%\\178.38\%\\12621.85\%\end{pmatrix}$
BM4018S118	$\begin{pmatrix}735.84\\736.03\\635.44\\521.89\\0.62\end{pmatrix} \pm \begin{pmatrix}0.09\%\\0.08\%\\0.06\%\\0.02\%\\0.14\%\end{pmatrix}$	$\begin{pmatrix}1933.84\\1934.35\\635.44\\521.89\\1.63\end{pmatrix} \pm \begin{pmatrix}0.31\%\\0.29\%\\0.06\%\\0.02\%\\0.37\%\end{pmatrix}$	$\begin{pmatrix}686.06\\686.22\\635.45\\521.92\\0.90\end{pmatrix} \pm \begin{pmatrix}0.10\%\\0.09\%\\0.06\%\\0.01\%\\0.09\%\end{pmatrix}$	$\begin{pmatrix}565.58\\565.68\\635.52\\521.81\\-0.23\\0.55\end{pmatrix} \pm \begin{pmatrix}0.70\%\\0.71\%\\0.06\%\\0.02\%\\2.33\%\\0.32\%\end{pmatrix}$	$\begin{pmatrix}736.95\\737.08\\635.52\\521.81\\0.57\\1.11\end{pmatrix} \pm \begin{pmatrix}0.11\%\\0.10\%\\0.06\%\\0.02\%\\0.60\%\\0.75\%\end{pmatrix}$	$\begin{pmatrix}736.92\\737.05\\635.52\\521.81\\0.02\\0.00\end{pmatrix} \pm \begin{pmatrix}0.11\%\\0.10\%\\0.06\%\\0.02\%\\4.26\%\\15.24\%\end{pmatrix}$	$\begin{pmatrix}737.02\\737.15\\635.52\\521.81\\0.02\\0.01\\-0.00\\0.00\end{pmatrix} \pm \begin{pmatrix}0.11\%\\0.10\%\\0.06\%\\0.02\%\\5.20\%\\21.76\%\\78.35\%\\191.37\%\end{pmatrix}$

Table 4.3: Mean and standard deviation (in %) of intrinsic parameters computed on three different sequences for each lens. The results suggest that our formulation of the UCM (first column, compare Eq. 4.7) has smaller standard deviation compared to standard formulation [86] (second column), where changes to γ and ξ have significant effect on each other.

Chapter 5

Direct Visual-Inertial Odometry with Stereo Cameras

Authors	Vladyslav Usenko[1]	usenko@in.tum.de
	Jakob Engel [1]	engelj@in.tum.de
	Jörg Stückler [1]	stueckle@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1]Technische Universität München, Munich, Germany	

Publication: V. Usenko, J. Engel, J. Stückler and D. Cremers. "Direct Visual-Inertial Odometry with Stereo Cameras". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. DOI: 10.1109/icra.2016.7487335

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 propose a novel direct visual-inertial odometry method for stereo cameras. Camera pose, velocity and IMU biases are simultaneously estimated by minimizing a combined photometric and inertial energy functional. This allows us to exploit the complementary nature of vision and inertial data. At the same time, and in contrast to all existing visual-inertial methods, our approach is fully direct: geometry is estimated in the form of semi-dense depth maps instead of manually designed sparse keypoints. Depth information is obtained both from static stereo – relating the fixed-baseline images of the stereo camera – and temporal stereo – relating images from the same camera, taken at different points in time. We show that our method outperforms not only vision-only or loosely coupled approaches, but also can achieve more accurate results than state-of-the-art keypoint-based methods on different datasets, including rapid motion and significant illumination changes. In addition, our method provides high-fidelity semi-dense, metric reconstructions of the environment, and runs in real-time on a CPU.

5.1 Introduction

Camera motion estimation and 3D reconstruction are amongst the most prominent topics in computer vision and robotics. They have major practical applications, well-known examples are robot navigation [130] [87] [123], autonomous or semi-autonomous driving [45], large-scale indoor reconstruction, virtual or augmented reality [89], and many more. In all of these scenarios, in the end one requires both the camera motion as well as information about the 3D structure of the environment – for example to recognize and navigate around obstacles, or to display environment-related information to a user.

In this paper, we propose a tightly coupled, direct visual-inertial stereo odometry. Combining a stereo camera with an inertial measurement unit (IMU), the method estimates accurate camera motion as well as semi-dense 3D reconstructions in real-time. Our approach combines two recent trends: *Direct image alignment* based on *probabilistic, semi-dense depth estimation* provides rich information about the environment, and allows for exploiting all information present in the images. This is in contrast to traditional feature-based approaches, which rely on hand-crafted keypoint detectors and descriptors, only utilizing information contained at, e.g., image corners – neglecting large parts of the image. Simultaneously, *tight integration* of inertial data into tracking provides accurate short-term motion constraints. This is of particular benefit for direct approaches: Direct image alignment is well-known to be heavily non-convex, and convergence can only be expected if a sufficiently accurate initial estimate is available. While in practice techniques like coarse-to-fine tracking increase the convergence radius, tight inertial integration solves this issue even more

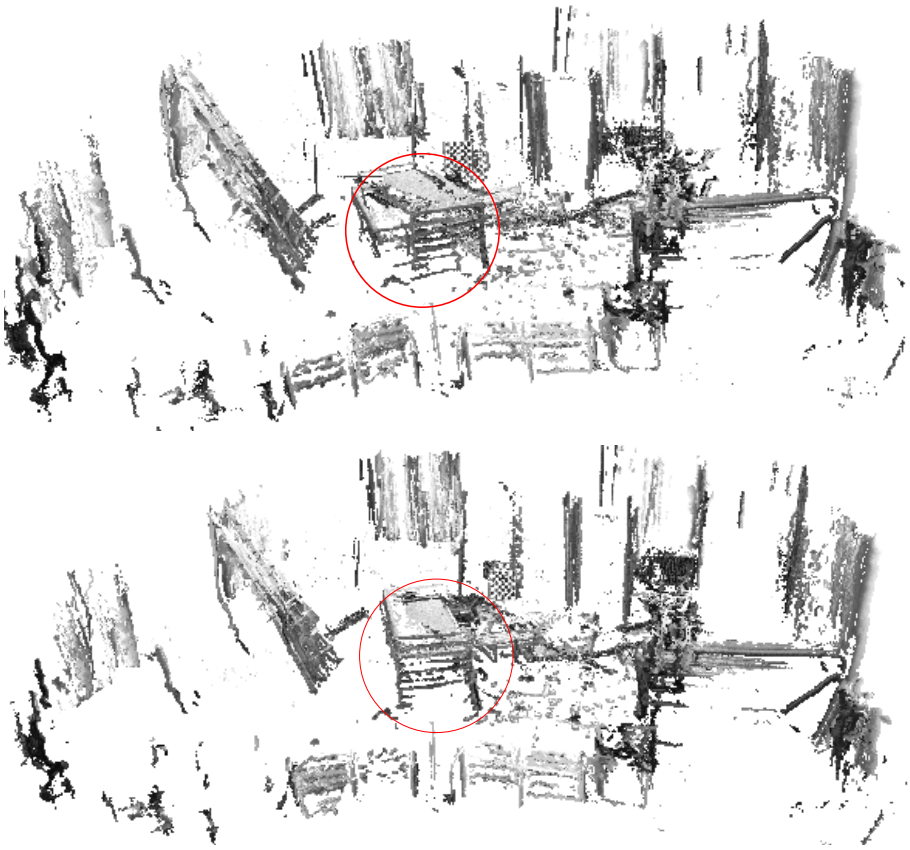


Figure 5.1: Tight fusion of the IMU measurements with direct image alignment results in more accurate position tracking (bottom) compared to the odometry system that only relies on image alignment (top). The reconstructed pointclouds come from pure odometry, no loop closures were enforced.

effectively, as the additional error term and resulting prior ensure convergence even for rapid motion. We show that it even allows for tracking through short intervals without visual information, e.g. caused by pointing the camera at a white wall. In addition, inertial measurements render global roll- and pitch observable, reducing global drift to translational 3D motion and yaw rotation.

In experiments we demonstrate the benefits of tight IMU integration with our Stereo LSD-SLAM approach [34] towards loose integration or vision-only approaches. Our method performs very well on challenging sequences with strong illumination changes and rapid motion. We also compare our method with state-of-the-art keypoint-based methods and demonstrate that our method can achieve better accuracy on challenging sequences.

5.2 Related Work

There exists a vast amount of research towards monocular and stereo visual odometry, 3D reconstruction and visual-inertial integration. In this section we will give an overview over the most relevant related publications, in particular focussing on direct vs. keypoint-based approaches, as well as tight vs. loosely coupled IMU integration.

While direct methods have a long history – first works including the work of Irani et al. [55] for monocular structure-and-motion – the first complete, real-time capable direct stereo visual-odometry was the work of Comport et al. [27]. Since then, direct methods have been omni-present in the domain of RGB-D cameras [66] [96], as they directly provide the required pixel-wise depth as sensor measurement. More recently, direct methods have become popular also in a monocular environment, prominent examples include DTAM [97], SVO [40] and LSD-SLAM [33].

At the same time, much progress has been made in the domain of IMU integration: due to their complementary nature and abundant presence in all modern hardware set-ups, IMUs are well-suited to complement vision-based systems – providing valuable information about short-term motion and rendering global roll, pitch, and scale observable. In early works, visual-inertial fusion has been approached as a pure sensor-fusion problem: Vision is treated as an independent, black-box 6-DoF sensor which is fused with inertial measurements in a filtering framework [130] [87] [35]. This so-called *loosely coupled* approach allows to use existing vision-only methods – such as PTAM [68], or LSD-SLAM [33] – without modifications; and the chosen method can easily be substituted for another one. On the other hand, in this approach, the vision part does not benefit from the availability of IMU data. More recent works therefore follow a tightly coupled approach, treating visual-inertial odometry as one integrated estimation problem, optimally exploiting both sensor modalities.

Two main categories can be identified: Filtering-based approaches [74] [17] [126] operate on a probabilistic state representation – mean and covariance – in a Kalman-filtering framework. One of the filtering approaches [48] claims to combine IMU measurements with direct image tracking, but does not provide a systematic evaluation and comparison to the state-of-the-art methods. Optimization-based approaches on the other hand operate on an energy-function based representation in a non-linear optimization framework. While the complementary nature of these two approaches has long been known [37], the energy-based approach [73] [58] [64], – which we employ in this paper – allows for easily and adaptively re-linearizing energy terms if required, thereby avoiding systematic error integration from linearization. Another example of energy-based approaches is presented in [39] which combines IMU measurements with direct tracking of a sparse subset of points in the image. In contrary to our method, old states are not marginalized out which on one hand allows for loop closures, but on the other hand does not guarantee bounded update time in

the worst case.

5.3 Contribution.

The main novelty of this paper is the formulation of *tight* IMU integration into *direct* image alignment within a non-linear energy-minimization framework. We show that including this sensor modality which in most practical cases is abundantly available helps to overcome the non-convexity of the photometric error, thereby eliminating one of the main weaknesses of direct approaches over keypoint-based methods. We evaluate our approach on different datasets and compare it to alternative stereo visual-inertial odometry systems, out-performing state-of-the-art keypoint-based methods in terms of accuracy in many cases. In addition, our method estimates accurate, metrically scaled, semi-dense 3D reconstructions of the environment, while running in real-time on a modern CPU.

5.4 Notation

Throughout the paper, we will write matrices as bold capital letters ($\mathbf{R}$) and vectors as bold lower case letters ($\boldsymbol{\xi}$). We will represent rigid-body poses directly as elements of $\mathfrak{se}(3)$, which – with a slight abuse of notation – we write directly as vectors, i.e., $\boldsymbol{\xi} \in \mathbb{R}^6$. We then define the pose concatenation operator $\circ\colon \mathfrak{se}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$ directly on this notation as $\boldsymbol{\xi} \circ \boldsymbol{\xi}' := \log\left(\exp(\boldsymbol{\xi})\exp(\boldsymbol{\xi}')\right)$.

For each time-step i, our method estimates the camera's rigid-body pose $\boldsymbol{\xi}_i \in \mathfrak{se}(3)$, its linear velocity $\boldsymbol{v}_i \in \mathbb{R}^3$ expressed in the world coordinate system, and the IMU bias terms $\boldsymbol{b}_i \in \mathbb{R}^6$ for the 3D acceleration and 3D rotational velocity measurements of the IMU. A full state is hence given by $\boldsymbol{s}_i := \left[\boldsymbol{\xi}_i^T \boldsymbol{v}_i^T \boldsymbol{b}_i^T\right]^T \in \mathbb{R}^{15}$. For ease of notation, we further define pose concatenation and subtraction directly on this state-space as

$$\boldsymbol{s}_i \oplus \boldsymbol{s}'_i := \begin{bmatrix} \boldsymbol{\xi}_i \circ \boldsymbol{\xi}'_i \\ \boldsymbol{v}_i + \boldsymbol{v}'_i \\ \boldsymbol{b} + \boldsymbol{b}'_i \end{bmatrix} \tag{5.1}$$

and

$$\boldsymbol{s}_i \ominus \boldsymbol{s}'_i := \begin{bmatrix} \boldsymbol{\xi}_i \circ \boldsymbol{\xi}_{\boldsymbol{i}}'^{-1} \\ \boldsymbol{v}_i - \boldsymbol{v}'_i \\ \boldsymbol{b} - \boldsymbol{b}'_i \end{bmatrix}. \tag{5.2}$$

The full state vector $\boldsymbol{s} := \left[\boldsymbol{s}_1^T \dots \boldsymbol{s}_N^T\right]^T$ includes the states of all frames.

5.5 Direct Visual-Inertial Stereo Odometry

We tightly couple direct image alignment – minimization of the photometric error – with non-linear error terms arising from inertial integration. In contrast to a loosely coupled approach, where the vision system runs independently of the IMU and is only fused afterwards, such tight integration maintains correlations between all state variables and thereby arbitrates directly between visual and IMU measurements.

Our photometric error formulation is directly based on the formulation proposed in LSD-SLAM [33], including robust Huber weights and normalization by the propagated depth variances. Recently, we extended this approach to stereo cameras [34] and augmented it with affine lighting correction. We then formulate a joint optimization problem to recover the full state containing camera pose, translational velocity and IMU biases of all frames i. The overall energy that we want to minimize is given by

$$E(\boldsymbol{s}) := \frac{1}{2}\sum_{i=1}^{N} E^{I}_{i\to\mathrm{ref}(i)}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_{\mathrm{ref}(i)}) + \frac{1}{2}\sum_{i=2}^{N} E^{\mathrm{IMU}}(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i), \tag{5.3}$$

where $E^{I}_{i\to\mathrm{ref}(i)}$ and E^{IMU}_{i} are image and IMU error function terms, respectively. This optimization problem can be interpreted as maximum-a-posterior estimation in a probabilistic graphical model (s. Fig. 5.2).

To achieve real-time performance, we do not optimize over an unboundedly growing number of state variables. Instead, we marginalize out all state variables other than the current image, its predecessor, and its reference keyframe. Through marginalization, all prior estimates and measurements are included with their uncertainty in the optimization.

Note that both modalities complement each other very well in a joint optimization framework – beyond the level of simple averaging of their motion estimates: Images can provide rich information for robust visual tracking. Depending on the observed scene, full 6-DoF relative motions can be observable. Degenerate cases, however, can occur in which the observed scene does not provide sufficient information for fully constrained tracking (e.g. pointing the camera at a texture-less wall). In this case, IMU measurements provide complementary measurements that bridge the gaps in observability.

IMUs typically operate at a much higher frequency than the frame-rate of the camera and make measuring gravity direction and eliminating drift in roll and pitch angles possible. The downside of IMUs is, however, that they measure relative poses only indirectly through rotational velocities and linear accelerations. They are noisy and need to be integrated and compensated for gravity which strongly depends on the accuracy of the pose estimate. The measurements come with unknown, drifting biases that need to be estimated using an external reference such as vision. While IMU information is incremental, any images can be aligned towards each other that

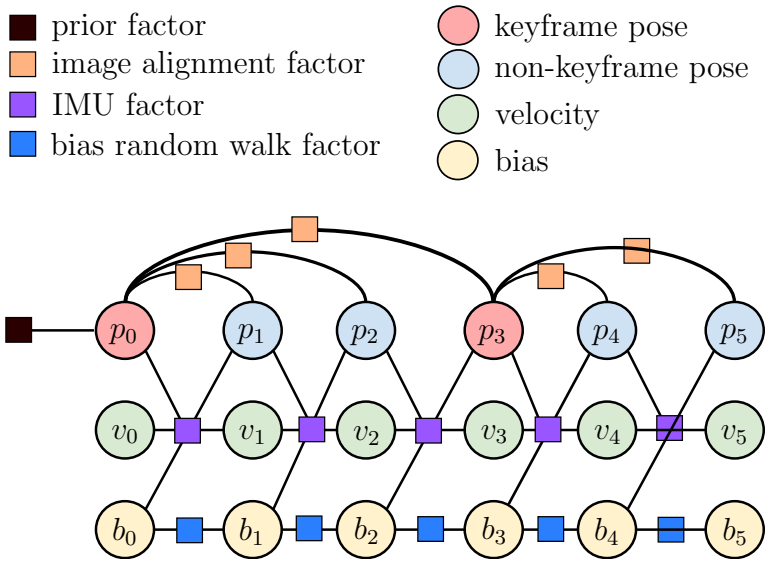


Figure 5.2: Factor graph representing the visual-inertial odometry optimization problem. Poses of the keyframes are shown in red, poses of other frames in blue, velocities in green and biases in yellow. Poses and velocities are connected to the pose, velocity and biases of the previous frame by an IMU factor. The pose of each frame is connected to the pose of the keyframe by a VO factor, and factors between biases constrain their random walk.

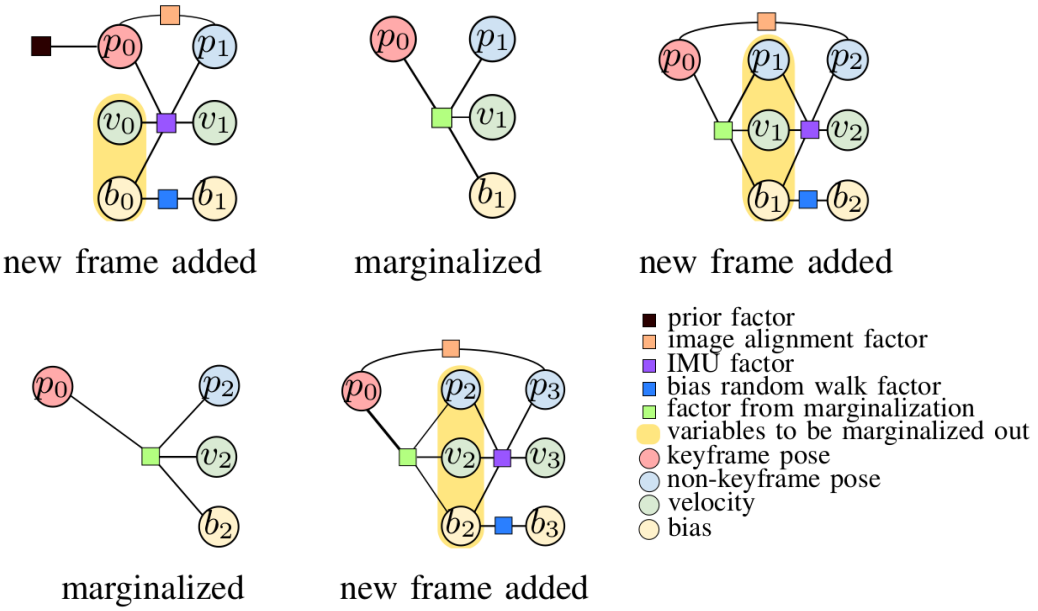


Figure 5.3: Evolution of the factor graph during tracking. After adding a new frame and optimizing the estimates of the variables in the graph, all variables except the keyframe pose and the current frame pose, velocity and biases are marginalized out. This process is repeated with each new frame.

have sufficient overlap. This allows for incorporating relative pose measurements between images that are not in direct temporal sequence–enabling more consistent trajectory estimates.

5.5.1 Direct Semi-Dense Stereo Odometry

We base visual tracking on Stereo LSD-SLAM [34]:

- We track the motion of the camera towards a reference keyframe in the map. We create new keyframes, if the camera moved too far from existing keyframes.
- We estimate a semi-dense depth map in the current reference keyframe from static and temporal stereo cues. For static stereo we exploit the fixed baseline between the pair of cameras in the stereo configuration. Temporal stereo is estimated from pixel correspondences in the reference keyframe towards subsequent images based on the tracked motion.

There are several benefits of complementing static with temporal stereo in a tracking and mapping framework. Static stereo makes reconstruction scale observable. It is also independent of camera movement, but is constrained to a constant baseline, which limits static stereo to an effective operating range. Temporal stereo requires non-degenerate camera movement, but is not bound to a specific range

as demonstrated in [33]. The method can reconstruct very small and very large environments at the same time. Finally, through the combination of static with temporal stereo, multiple baseline directions are available: while static stereo typically has a horizontal baseline – which does not allow for estimating depth along horizontal edges, temporal stereo allows for completing the depth map by providing other motion directions.

Direct Image Alignment

The pose between two images I_1 and I_2 is estimated by minimizing the photometric residuals

$$r^I_{\boldsymbol{u}}(\boldsymbol{\xi}) := aI_1(\boldsymbol{u}) + b - I_2(\boldsymbol{u}'). \tag{5.4}$$

where $\boldsymbol{u}' := \pi\left(\mathbf{T}_{\boldsymbol{\xi}}\pi^{-1}\left(\boldsymbol{u}, D_1(\boldsymbol{u})\right)\right)$ and $\boldsymbol{\xi}$ transforms from image frame I_2 to I_1. The mappings π and π^{-1} project points from the image to the 3D domain and vice versa using a pinhole camera model. The parameters a and b correct for affine lighting changes between the images and are optimized alongside the pose $\boldsymbol{\xi}$ in an alternating fashion, as described in [34]. We also determine the uncertainty $\sigma^I_{r,\boldsymbol{u}}$ of this residual [33]. The optimization objective for tracking a current frame towards a keyframe is thus given by

$$E^I_{\text{cur}\to\text{ref}}(\boldsymbol{\xi}_{\text{cur}}, \boldsymbol{\xi}_{\text{ref}}) := \sum_{u\in\Omega_{D_1}} \rho\left(\left[\frac{r^I_{\boldsymbol{u}}(\boldsymbol{\xi}^{-1}_{\text{ref}} \circ \boldsymbol{\xi}_{\text{cur}})}{\sigma^I_{r,\boldsymbol{u}}}\right]^2\right), \tag{5.5}$$

where ρ is the Huber norm. This objective is minimized using the iteratively reweighted Levenberg-Marquardt algorithm.

Depth Estimation

Scene geometry is estimated for pixels of the key frame with high image gradient, since they provide stable disparity estimates. Fig. 5.1 shows an example of such a semi-dense depth reconstruction. We estimate depth both from static stereo (i.e., using images from different physical cameras, but taken at the same point in time) as well as from temporal stereo (i.e., using images from the same physical camera, taken at different points in time).

We initialize the depth map with the propagated depth from the previous keyframe. The depth map is subsequently updated with new observations in a pixel-wise depth-filtering framework. We also regularize the depth maps spatially and remove outliers [33].

Static Stereo We determine the static stereo disparity at a pixel by a correspondence search along its epipolar line in the other stereo image. In our case of stereo-rectified images, this search can be performed very efficiently along horizontal lines. We use the SSD photometric error over five pixels along the scanline as a correspondence measure. If a depth estimate with uncertainty is available, the search range along the epipolar line can be significantly reduced. Due to the fixed baseline, we limit disparity estimation to pixels with significant gradient along the epipolar line, making the depth reconstruction semi-dense.

Static stereo is integrated in two ways. If a new stereo keyframe is created, the static stereo in this keyframe stereo pair is used to initialize the depth map. During tracking, static stereo in the current frame is propagated to the reference frame and fused with its depth map.

Temporal Stereo For temporal stereo we estimate disparity between the current frame and the reference keyframe using the pose estimate obtained through tracking. These estimates are fused in the keyframe. Only pixels are updated with temporal stereo, whose expected inverse depth error is sufficiently small. This also constrains depth estimates to pixels with high image gradient along the epipolar line, producing a semi-dense depth map.

5.5.2 IMU Integration

Underlying our IMU error function terms is the following nonlinear dynamical model. Let the pose $\boldsymbol{\xi}$ consist of the position $\boldsymbol{p}$ and rotation $\mathbf{R}$ of the IMU expressed in the world frame. Note that the velocity estimate $\boldsymbol{v}$ also is in the world frame. According to the IMU measurements of rotational velocities $\boldsymbol{\omega}_z$ and linear accelerations $\boldsymbol{a}_z$ the pose of the IMU evolves as

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{5.6}$$

$$\dot{\boldsymbol{v}} = \mathbf{R}\left(\boldsymbol{a}_z + \boldsymbol{\epsilon}_a - \boldsymbol{b}_a\right) + \boldsymbol{g} \tag{5.7}$$

$$\dot{\mathbf{R}} = \mathbf{R}\left[\boldsymbol{\omega}_z + \boldsymbol{\epsilon}_\omega - \boldsymbol{b}_\omega\right]_\times \tag{5.8}$$

where $[\cdot]_\times$ is the skew-symmetric matrix such that for vectors a, b, $[a]_\times b = a \times b$. The process noise $\boldsymbol{\epsilon}_a$, $\boldsymbol{\epsilon}_\omega$, $\boldsymbol{\epsilon}_{b,a}$, and $\boldsymbol{\epsilon}_{b,\omega}$ affect the measurements and their biases $\boldsymbol{b}_a$ and $\boldsymbol{b}_\omega$ with Gaussian white noise. Hence, for the biases $\dot{\boldsymbol{b}}_a = \boldsymbol{\epsilon}_{b,a}$ and $\dot{\boldsymbol{b}}_\omega = \boldsymbol{\epsilon}_{b,\omega}$. Note that we neglect the effect of Coriolis force in this model.

IMU measurements typically arrive at a much higher frequency than camera frames. We do not add independent residuals for each individual IMU measurement, but integrate the measurements into a condensed IMU measurement between the image frames. In order to avoid frequent reintegration if the pose or bias estimates change during optimization, we follow the pre-integration approach proposed in [79]

and [54]. We integrate the IMU measurements between timestamps i and j in the IMU coordinate frame and obtain pseudo-measurements $\Delta\boldsymbol{p}_{i\to j}$, $\Delta\boldsymbol{v}_{i\to j}$, and $\mathbf{R}_{i\to j}$.

We initialize pseudo-measurements with $\Delta\boldsymbol{p}_{i\to i} = 0$, $\Delta\boldsymbol{v}_{i\to i} = 0$, $\mathbf{R}_{i\to i} = \mathbf{I}$, and assuming the time between IMU measurements is Δt we integrate the raw measurements:

$$\Delta\boldsymbol{p}_{i\to k+1} = \Delta\boldsymbol{p}_{i\to k} + \Delta\boldsymbol{v}_{i\to k}\Delta t \tag{5.9}$$

$$\Delta\boldsymbol{v}_{i\to k+1} = \Delta\boldsymbol{v}_{i\to k} + \mathbf{R}_{i\to k}\left(\boldsymbol{a}_z - \boldsymbol{b}_a\right)\Delta t \tag{5.10}$$

$$\mathbf{R}_{i\to k+1} = \mathbf{R}_{i\to k}\exp([\boldsymbol{\omega}_z - \boldsymbol{b}_\omega]_\times\,\Delta t). \tag{5.11}$$

Given the initial state and integrated measurements the state at the next time-step can be predicted:

$$\boldsymbol{p}_j = \boldsymbol{p}_i + (t_j - t_i)\boldsymbol{v}_i + \frac{1}{2}(t_j - t_i)^2\boldsymbol{g} + \mathbf{R}_i\Delta\boldsymbol{p}_{i\to j} \tag{5.12}$$

$$\boldsymbol{v}_j = \boldsymbol{v}_i + (t_j - t_i)\boldsymbol{g} + \mathbf{R}_i\Delta\boldsymbol{v}_{i\to j} \tag{5.13}$$

$$\mathbf{R}_j = \mathbf{R}_i\mathbf{R}_{i\to j}. \tag{5.14}$$

For the previous state $\boldsymbol{s}_{i-1}$ and IMU measurements $\boldsymbol{a}_{i-1}$, $\boldsymbol{\omega}_{i-1}$ between frames i and $i-1$, the method yields a prediction

$$\widehat{\boldsymbol{s}}_i := h\left(\boldsymbol{\xi}_{i-1}, \boldsymbol{v}_{i-1}, \boldsymbol{b}_{i-1}, \boldsymbol{a}_{i-1}, \boldsymbol{\omega}_{i-1}\right) \tag{5.15}$$

of the pose, velocity, and biases in frame i with associated covariance estimate $\widehat{\Sigma}_{s,i}$. Hence, the IMU error function terms are

$$E^{\text{IMU}}(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i) := \left(\boldsymbol{s}_i \ominus \widehat{\boldsymbol{s}}_i\right)^T \widehat{\boldsymbol{\Sigma}}_{s,i}^{-1}\left(\boldsymbol{s}_i \ominus \widehat{\boldsymbol{s}}_i\right). \tag{5.16}$$

5.5.3 Optimization

The error function in eq. (5.3) can be written as

$$E(\boldsymbol{s}) = \frac{1}{2}\boldsymbol{r}^T\mathbf{W}\boldsymbol{r} \tag{5.17}$$

$$= \frac{1}{2}\begin{bmatrix}\boldsymbol{r}_I^T & \boldsymbol{r}_{\text{IMU}}^T\end{bmatrix}\begin{bmatrix}\mathbf{W}_I & 0\\ 0 & \mathbf{W}_{\text{IMU}}\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_I\\ \boldsymbol{r}_{\text{IMU}}\end{bmatrix}. \tag{5.18}$$

The weights either implement the Huber norm on the photometric residuals $\boldsymbol{r}_I$ using iteratively re-weighted least-squares, or correspond to the inverse covariances of the IMU residuals $\boldsymbol{r}_{\text{IMU}}$ (eq.(5.16)). We optimize this objective using the Levenberg-Marquardt method. Linearizing the residual around the current state

$$\boldsymbol{r}(\boldsymbol{s} \oplus \boldsymbol{\delta s}) = \boldsymbol{r}(\boldsymbol{s}) + \mathbf{J}_s\boldsymbol{\delta s} \tag{5.19}$$

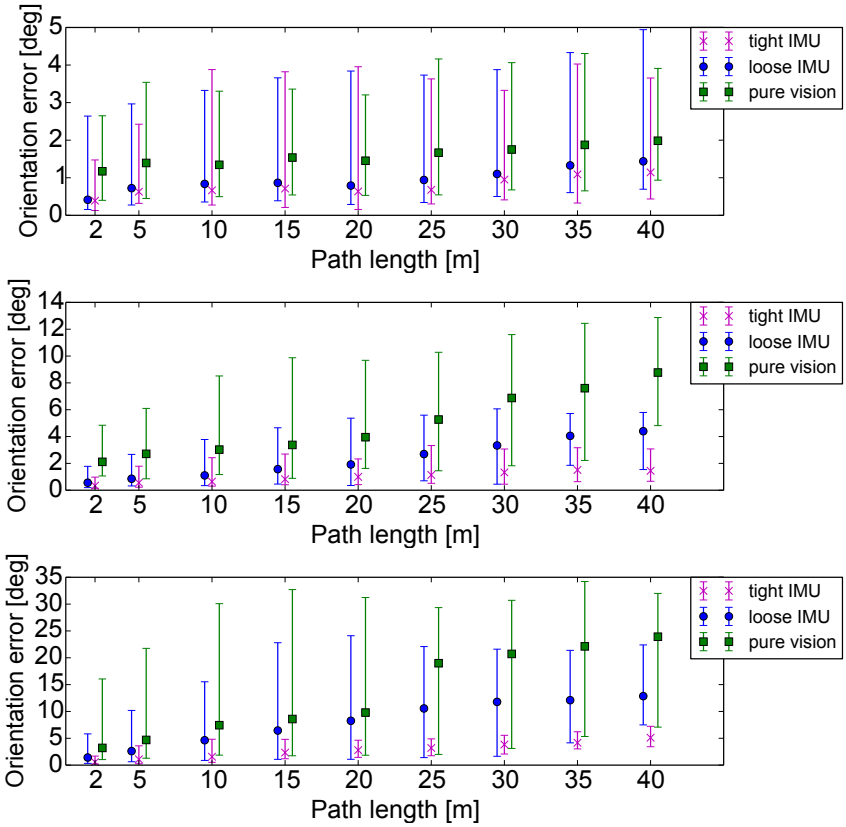


Figure 5.4: Orientation error evaluated over different segment lengths for the three EuRoC dataset sequences (from top to bottom: stage 1 task 2.1, 2.2, 2.3). While both loosely-coupled and tightly-coupled IMU integration significantly decrease the error as global roll and pitch become observable, the tightly coupled approach is clearly superior. In particular in the last sequence – which includes strong motion blur and illumination changes – direct tracking directly benefits from tight IMU integration. See also Fig. 5.5.

where

$$\mathbf{J}_s = \left.\frac{d\boldsymbol{r}(\boldsymbol{s} \oplus \boldsymbol{\delta s})}{d\boldsymbol{\delta s}}\right|_{\boldsymbol{\delta s}=\mathbf{0}}, \tag{5.20}$$

the error function $E(\boldsymbol{s})$ can be approximated around the current state $\boldsymbol{s}$ with a quadratic function

$$E(\boldsymbol{s} \oplus \boldsymbol{\delta s}) = E_{\boldsymbol{s}} + \boldsymbol{\delta s}^T \boldsymbol{b}_{\boldsymbol{s}} + \frac{1}{2}\boldsymbol{\delta s}^T \mathbf{H}_{\boldsymbol{s}} \boldsymbol{\delta s} \tag{5.21}$$

$$\boldsymbol{b}_{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}^T \mathbf{W} \boldsymbol{r}(\boldsymbol{s}) \tag{5.22}$$

$$\mathbf{H}_{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}^T \mathbf{W} \mathbf{J}_{\boldsymbol{s}} \tag{5.23}$$

where $\boldsymbol{b}_{\boldsymbol{s}}$ is the Jacobian and $\mathbf{H}_{\boldsymbol{s}}$ is the Hessian approximation of $E(\boldsymbol{s})$ and $\boldsymbol{\delta s}$ is a right-multiplied increment to the current state. This function is minimized through $\boldsymbol{\delta s} = -\mathbf{H}_{\boldsymbol{s}}^{-1}\boldsymbol{b}_{\boldsymbol{s}}$, yielding the state update $\boldsymbol{s} \leftarrow \boldsymbol{s} \oplus \boldsymbol{\delta s}$. This update and relinearization process is repeated until convergence.

5.5.4 Partial Marginalization

To constrain the size of the optimization problem, we perform partial marginalization and keep the set of optimized states at a small constant size. Specifically, we only optimize for the current frame state $\boldsymbol{s}_i$, the state of the previous frame $\boldsymbol{s}_{i-1}$, and the state $\boldsymbol{s}_{\text{ref}}$ of the reference frame used for tracking. If we split our state space $\boldsymbol{s}$ into $\boldsymbol{s}_\lambda$ and $\boldsymbol{s}_\mu$, where $\boldsymbol{s}_\lambda$ are the state variables we want to keep in the optimization, and $\boldsymbol{s}_\mu$ are the parts of the state that we want to marginalize out, we can rewrite the update step as follows

$$\begin{bmatrix} \mathbf{H}_{\mu\mu} & \mathbf{H}_{\mu\lambda} \\ \mathbf{H}_{\lambda\mu} & \mathbf{H}_{\lambda\lambda} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta s}_\mu \\ \boldsymbol{\delta s}_\lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_\mu \\ \boldsymbol{b}_\lambda \end{bmatrix}. \tag{5.24}$$

Applying the Schur complement to the upper part of the system we find

$$\boldsymbol{\delta s}_\lambda = -\left(\mathbf{H}_{\lambda\lambda}^*\right)^{-1} \boldsymbol{b}_\lambda^*, \tag{5.25}$$

$$\mathbf{H}_{\lambda\lambda}^* = \mathbf{H}_{\lambda\lambda} - \mathbf{H}_{\lambda\mu}\mathbf{H}_{\mu\mu}^{-1}\mathbf{H}_{\mu\lambda}, \tag{5.26}$$

$$\boldsymbol{b}_\lambda^* = \boldsymbol{b}_\lambda - \mathbf{H}_{\lambda\mu}\mathbf{H}_{\mu\mu}^{-1}\boldsymbol{b}_\mu. \tag{5.27}$$

which represents a system for $E^*(\boldsymbol{s}_\lambda)$ with states $\boldsymbol{s}_\mu$ marginalized out. Figure 5.3 shows the evolution of the graph with the marginalization procedure applied after adding every new frame to the graph.

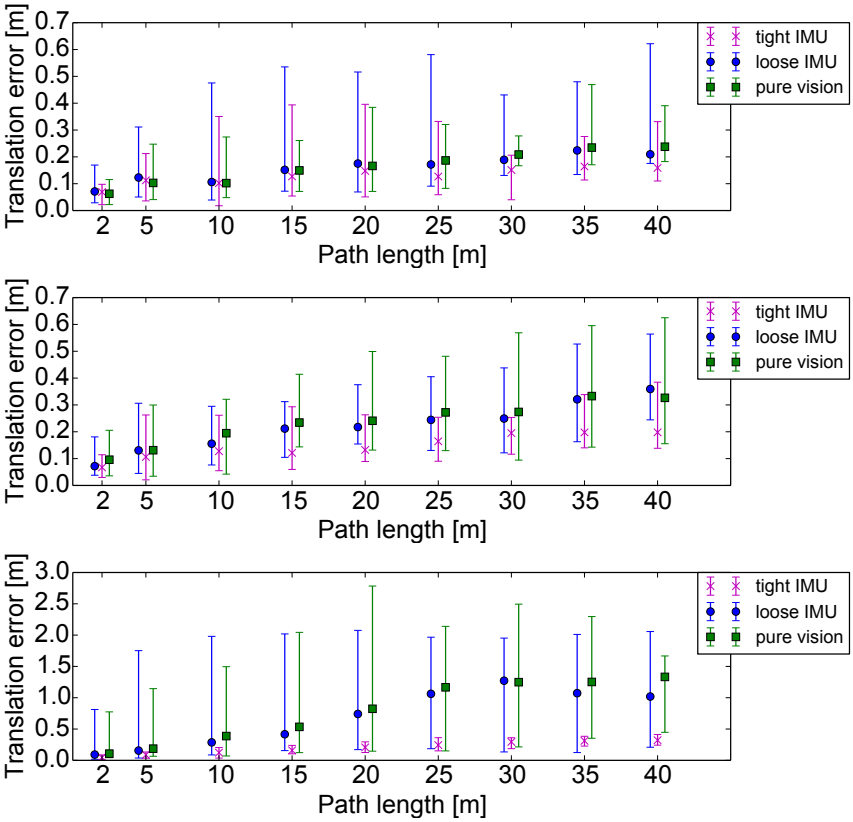


Figure 5.5: Translational drift evaluated over different segment lengths for the three EuRoC sequences (from top to bottom: stage 1 task 2.1, 2.2, 2.3). As for rotation (Fig. 5.4), the tightly coupled approach clearly performs best.

5.5.5 Changing the Linearization Point

Partial marginalization fixes the linearization point of $\boldsymbol{s}_\lambda$ for the quantities involving both $\boldsymbol{s}_\mu$ and $\boldsymbol{s}_\lambda$ in eq. (5.25). Further optimization, however, changes the linearization point such that a relinearization would be required. We avoid the tedious explicit relinearization using a first-order approximation. If we represent the new linearization point $\boldsymbol{s}'_\lambda$ by the old linearization point $\boldsymbol{s}_\lambda$ and an increment $\boldsymbol{\Delta s}_\lambda$,

$$\boldsymbol{s}'_\lambda = \boldsymbol{s}_\lambda \oplus \underbrace{\boldsymbol{s}_\lambda^{-1} \oplus \boldsymbol{s}'_\lambda}_{=:\boldsymbol{\Delta s}_\lambda}, \tag{5.28}$$

we can change the linearization point of the current quadratic approximation of E^* through

$$\begin{aligned} E^*(\boldsymbol{s}'_\lambda \oplus \boldsymbol{\delta s}_\lambda) &= E^*(\boldsymbol{s}_\lambda \oplus \boldsymbol{\Delta s}_\lambda \oplus \boldsymbol{\delta s}_\lambda) && (5.29)\\ &\approx E^*(\boldsymbol{s}_\lambda \oplus (\boldsymbol{\Delta s}_\lambda + \boldsymbol{\delta s}_\lambda)). && (5.30) \end{aligned}$$

The approximation made holds only if both $\boldsymbol{\Delta s}_\lambda$ and $\boldsymbol{\delta s}_\lambda$ are small – as both represent updates to the state, this is a valid assumption. We can then approximate the error function linearized around $\boldsymbol{s}'_\lambda$:

$$\begin{aligned} E^*(\boldsymbol{s}'_\lambda \oplus \boldsymbol{\delta s}_\lambda) &= E^*_{\lambda'} + \boldsymbol{\delta s}_\lambda^T \boldsymbol{b}^*_{\lambda'} + \frac{1}{2}\boldsymbol{\delta s}_\lambda^T \mathbf{H}^*_{\lambda'\lambda'} \boldsymbol{\delta s}_\lambda, && (5.31)\\ E^*_{\lambda'} &= E^*_\lambda + \frac{1}{2}\boldsymbol{\Delta s}_\lambda{}^T \mathbf{H}^*_{\lambda\lambda} \boldsymbol{\Delta s}_\lambda + \boldsymbol{\Delta s}_\lambda^T \boldsymbol{b}^*_\lambda, && (5.32)\\ \boldsymbol{b}^*_{\lambda'} &= \boldsymbol{b}^*_\lambda + \mathbf{H}^*_{\lambda\lambda} \boldsymbol{\Delta s}_\lambda, && (5.33)\\ \mathbf{H}^*_{\lambda'\lambda'} &= \mathbf{H}^*_{\lambda\lambda}. && (5.34) \end{aligned}$$

5.5.6 Statistical Consistency

Our framework accumulates information from many sources, in particular it uses (1) IMU observations, (2) static stereo, (3) temporal stereo / direct tracking and (4) a smoothness-prior on the depth. While old camera poses are correctly marginalized, we discard all pose-depth and depth-depth correlations: For each image alignment factor, depth values are treated as independent (noisy) input (eq. (5.5)). In turn, the effect of noisy poses is approximated during depth estimation [36]. While from a statistical perspective this is clearly incorrect, it allows our system to use hundreds of thousands of residuals per direct image alignment factor in real-time. Furthermore, it becomes unnecessary to drop past observations in order to preserve depth-depth independence, as done in [68]. Also note that – in contrast to monocular LSD-SLAM – much of the depth information originates from static stereo, which in fact is independent of the tracked camera poses.

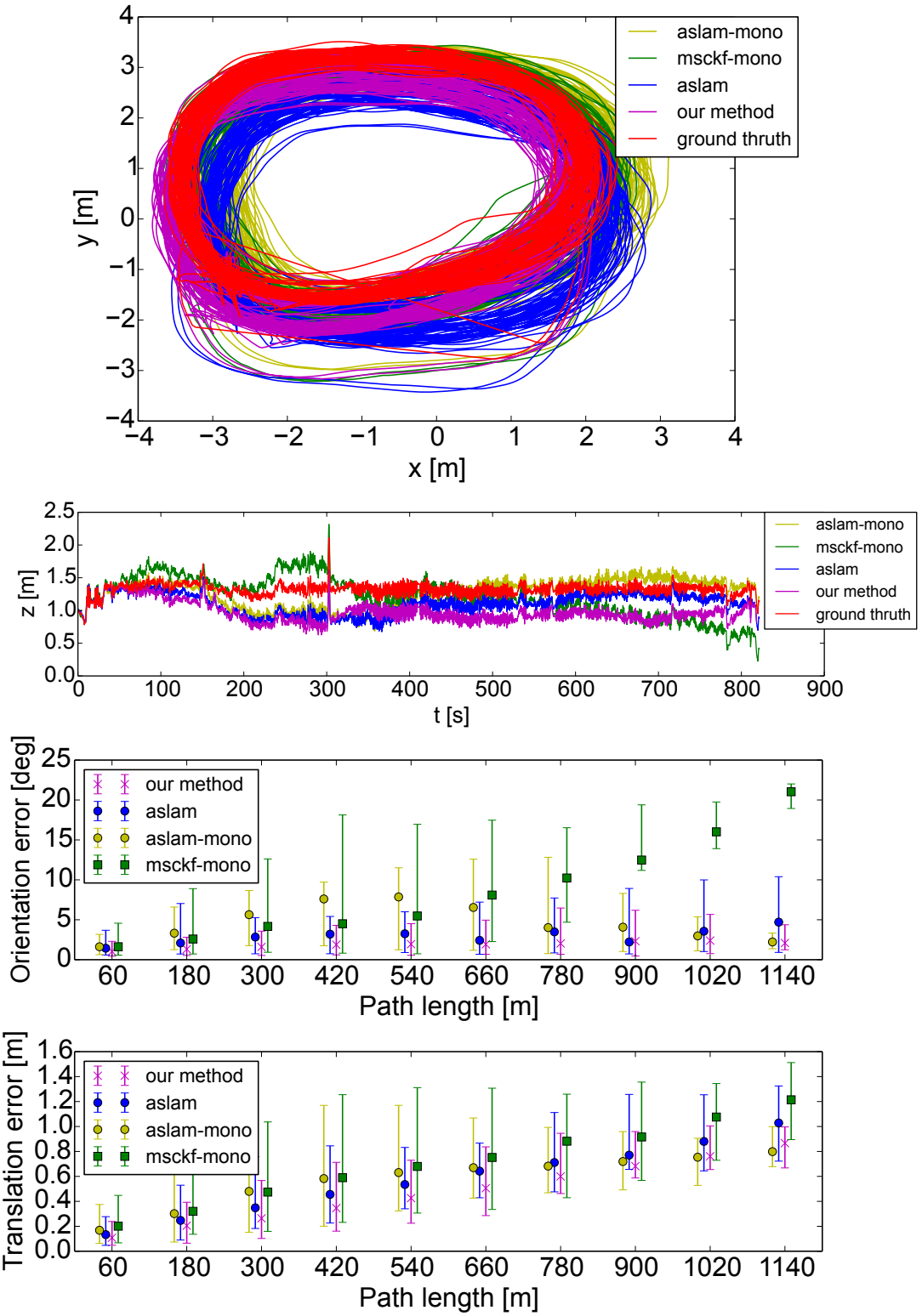


Figure 5.6: Long-run comparison with state-of-the-art keypoint-based VI odometry methods, both filtering-based (msckf) and optimization-based (aslam). Dataset and results reported in [73]. Top: horizontal trajectory plot. Middle: height estimate. Bottom: Translational/rotational drift evaluated over different segment lengths.

Figure 5.7: Qualitative results on a subset of Malaga Urban Dataset. Semi-dense reconstructions of selected parts of the map are shown on the left, and the overall trajectory with semi-dense reconstruction is shown in the middle. On the right a map of the city with overlayed trajectory measured with GPS is presented.

5.6 Results

We evaluate our approach both qualitatively and quantitatively on three different datasets, including a direct comparison to state-of-the-art feature-based VI odometry methods.

We have selected the datasets as they are especially challenging for direct visual odometry methods. They contain contrast changes, pure rotations, aggressive motions and relatively few frames per second, and for all of them the pure monocular algorithm [33] fails to track the sequence until the end. For Malaga dataset we used the default calibration parameters for evaluation. For all other datasets an offline calibration was performed using the *Kalibr* framework [43].

We compare loosely- with tightly-coupled IMU integration with Stereo LSD-SLAM. The loosely coupled version runs the direct image alignment process separately, and only the final pose estimation result of the alignment is included into the optimization as a relative pose constraint between reference and current frame. With regards to the reconstruction accuracy, ground truth is not available on the datasets, such that it can only be judged qualitatively. Nevertheless, as tracking is based on the reconstruction, its accuracy implicitly depends on the trajectory estimate.

5.6.1 EuRoC Dataset

This dataset is obtained from the European Robotics Challenge (EuRoC), and contains three calibrated stereo video sequences with corresponding IMU measurements,

recorded with a Skybotix VI sensor. They were obtained from a quadrocopter flying indoors (stage 1, tasks 2.1, 2.2, 2.3) and are in increasing difficulty: The third and most challenging sequence includes fast and aggressive motion, strong illumination changes as well as motion-blur and poorly textured views; some example images are shown in Fig. 5.8, as well as in the attached video. The images are provided with all required calibration parameters and motion-capture based ground truth, at WVGA resolution.

On this dataset, we evaluate the difference between tight IMU integration, loose IMU integration and purely vision-based LSD-SLAM. With the two upper plots in Fig. 5.6, we give a qualitative impression of the absolute trajectory estimate as in [73]. Since visual odometry does not correct for drift like a SLAM or full bundle adjustment method, the quantitative performance of the algorithm can be judged from the relative pose error (RPE) measure in the two bottom plots. The results in Fig. 5.5 and Fig. 5.4 demonstrate that our tightly-integrated, direct visual-inertial odometry method outperforms loose IMU integration both in translation and orientation drift. Both IMU methods in turn are better than the purely vision-based approach. The differences become particularly obvious for the last sequence, as here the tight IMU integration greatly helps to overcome non-convexities in the photometric error, allowing to seamlessly track through parts with strong motion blur.

Qualitatively, the reconstruction in Fig. 5.1 demonstrates a significant reduction in drift through tight IMU integration. The improved performance becomes apparent through the well-aligned, highlighted reconstructions which are viewed at the beginning and the end of the trajectory.

5.6.2 Long-Term Drift Evaluation

The second dataset contains a 14 minutes long sequence designed to evaluate long-term drift, captured with the same hardware setup as the EuRoC dataset. It is described and evaluated in [73] facilitating direct numeric comparison.

On this dataset, we compare our method with stereo depth estimation to the keypoint-based nonlinear optimization methods presented in [73] (aslam, aslam-mono) and the filtering-based approach in [92] (msckf). The aslam methods come in a stereo (aslam) and a monocular (aslam-mono) version. From Fig. 5.5 we can observe that the proposed method outperforms the filtering-based approach and the state-of-the-art keypoint-based optimization methods. Note, that our method at the same time provides a semi-dense 3D reconstruction of the environment.

5.6.3 Malaga Dataset: Autonomous Driving

Third, we provide qualitative results on the Malaga dataset [16], obtained from a car-mounted stereo camera. For this dataset only raw GPS position without ori-

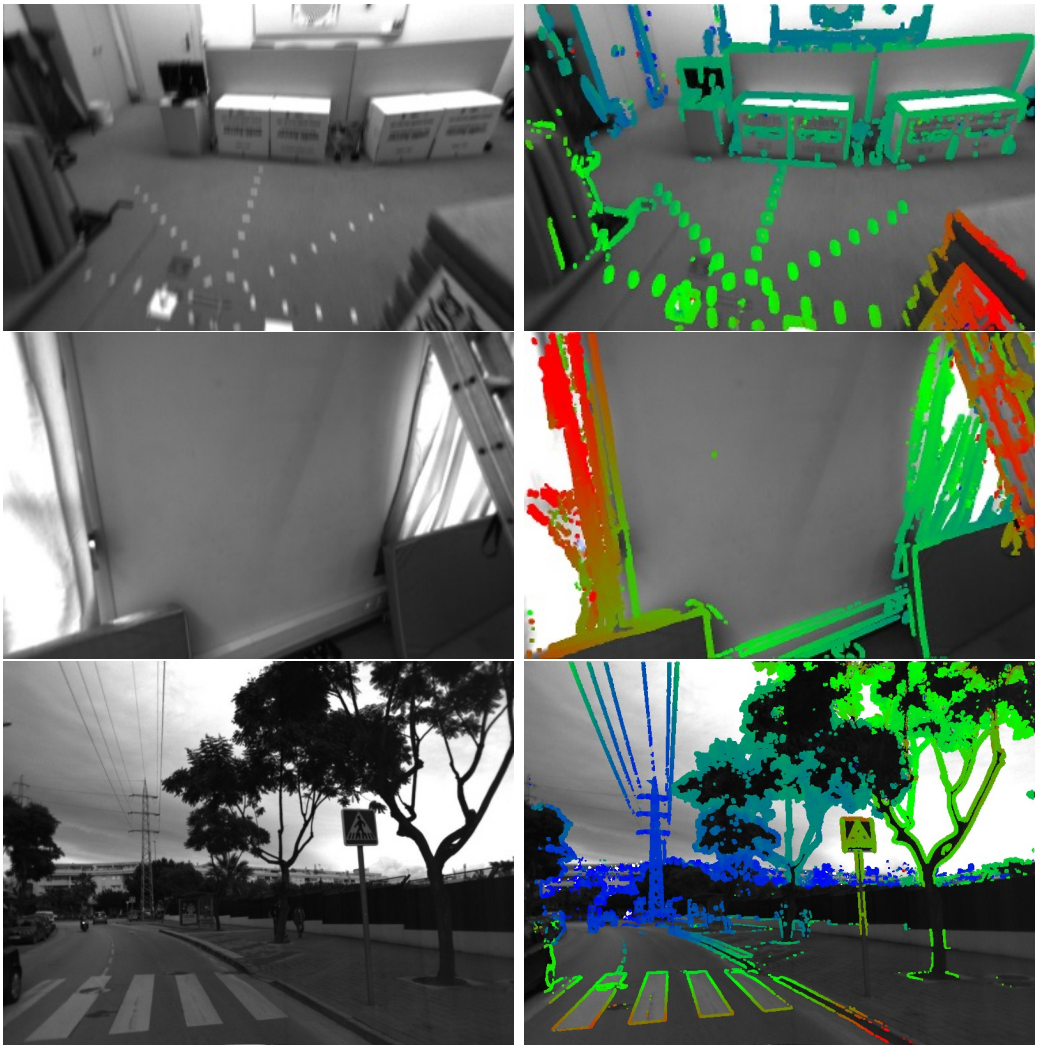


Figure 5.8: Images from the EuRoC (upper row: motion blur, middle row: textureless) and Malaga datasets (bottom row) with semi-dense depth estimates. Semi-dense depth maps, with color-coded depth estimates are shown on the left.

entation is available as ground-truth such that we cannot provide a quantitative evaluation. Figure 5.8 shows a resulting trajectory, a semi-dense reconstruction of the environment, and a city map overlaid with GPS signal obtained on the Malaga dataset. These qualitative results demonstrate our algorithm in a challenging outdoor application scenario. Repetitive texture, moving cars and pedestrians, and direct sunlight pose gross challenges to vision-based approaches.

5.7 Conclusion

We have presented a novel approach to direct, tightly integrated visual-inertial odometry. It combines a fully direct structure and motion approach – operating on per-pixel depth instead of individual keypoint observations – with tight, minimization-based IMU integration. We show that the two sensor sources ideally complement each other: stereo vision allows the system to compensate for long-term IMU bias drift, while short-term IMU constraints help to overcome non-convexities in the photometric tracking formulation, allowing to track through large inter-frame motion or intervals without visual information. Our method can outperform existing feature-based approaches in terms of tracking accuracy, and simultaneously provides accurate semi-dense 3D reconstructions of the environment, while running in real-time on a standard laptop CPU.

In future work, we will investigate tight IMU integration with monocular LSD-SLAM. We also plan to employ this technology for localization and mapping with flying and mobile robots as well as handheld devices.

Chapter 6

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@tum.de
	[1]Technische Universität München, Munich, Germany	

Publication: L. von Stumberg, V. Usenko and D. Cremers. "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. May 2018. DOI: 10.1109/ICRA.2018.8462905. eprint: http://arxiv.org/abs/1804.05625v1

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

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.

6.1 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 [66], 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 [45]. 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) [32] 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

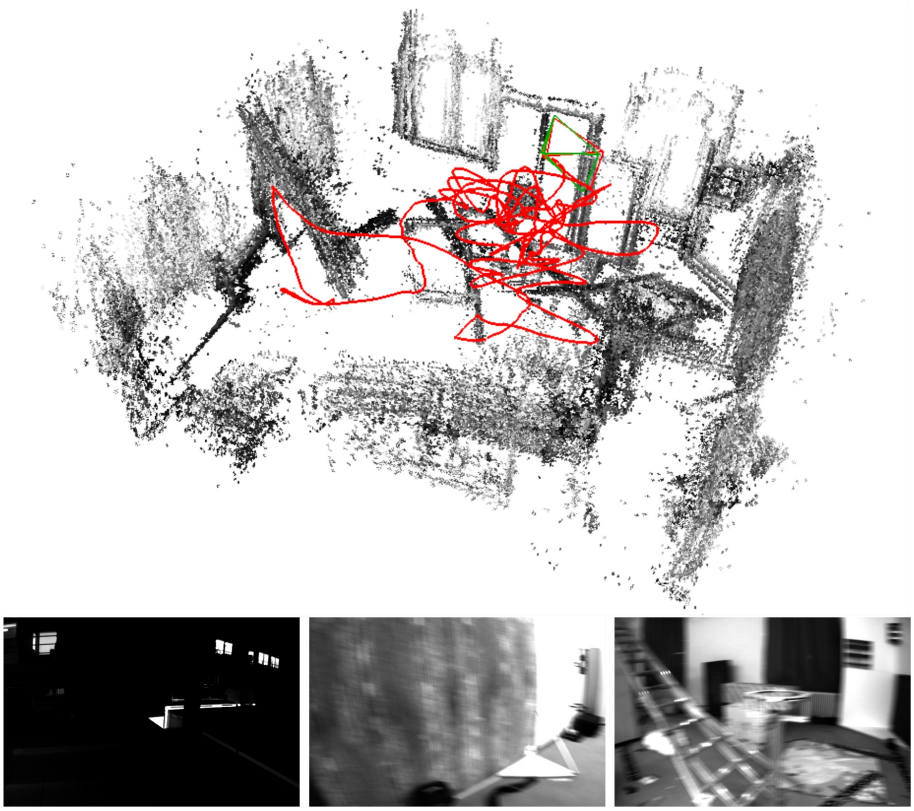


Figure 6.1: Bottom: Example images from the EuRoC-dataset: Low illumination, strong motion blur and little texture impose significant challenges for odometry estimation. Still our method is able to process all sequences with a rmse of less then 0.23m. Top: Reconstruction, estimated pose (red camera) and groundtruth pose (green camera) at the end of V1_03_difficult.

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.

Quantitative evaluation on the EuRoC dataset [20] 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. 6.1).

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.

6.2 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. [100], 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 [68], 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. [94].

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 [27]. Several methods for motion estimation for RGB-D cameras were developed by Kerl et al. [66]. More recently, direct approaches were also applied to monocular cameras, in a dense [97], semi-dense [33], and sparse fashion [40] [32].

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 [87] and [35]. 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 [74] [17] and energy-minimization based approaches [73] [39] [8] [95].

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 [85], and extended to consider IMU biases in [60].

6.3 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 [32] and can be viewed as a direct formulation of [73]. In contrast to [8], we jointly determine poses and 3D geometry from a single optimization function. This results in better precision especially on hard sequences. Compared to [39] 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

$$E_{\text{total}} = \lambda \cdot E_{\text{photo}} + E_{\text{inertial}} \tag{6.1}$$

which consists of the photometric error E_{photo} (section 6.3.2) and an inertial error term E_{inertial} (section 6.3.3).

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 [95] 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.

6.3.1 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.

6.3.2 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:

$$E_{\boldsymbol{p}j} = \sum_{\mathbf{p}\in\mathcal{N}_{\boldsymbol{p}}} \omega_{\boldsymbol{p}} \left\| (I_j[\boldsymbol{p'}] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right\|_\gamma, \tag{6.2}$$

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

$$E_{\text{photo}} = \sum_{i\in\mathcal{F}} \sum_{\boldsymbol{p}\in\mathcal{P}_i} \sum_{j\in\text{obs}(\boldsymbol{p})} E_{\boldsymbol{p}j}, \tag{6.3}$$

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.

6.3.3 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 [8, eq. (6), (7), (8)].

As IMU data is obtained with a much higher frequency than images we follow the preintegration approach proposed in [79] and improved in [24] and [39]. 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 (6.9)), 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

$$E_{\text{inertial}}(\boldsymbol{s}_i, \boldsymbol{s}_j) := (\boldsymbol{s}_j \boxminus \widehat{\boldsymbol{s}}_j)^T \, \widehat{\boldsymbol{\Sigma}}_{s,j}^{-1} \, (\boldsymbol{s}_j \boxminus \widehat{\boldsymbol{s}}_j) \tag{6.4}$$

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.

6.3.4 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 [85]. [85, 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 [95] uses the first 15 seconds of camera motion for the initialization on the EuRoC dataset to make sure that all values are observable.

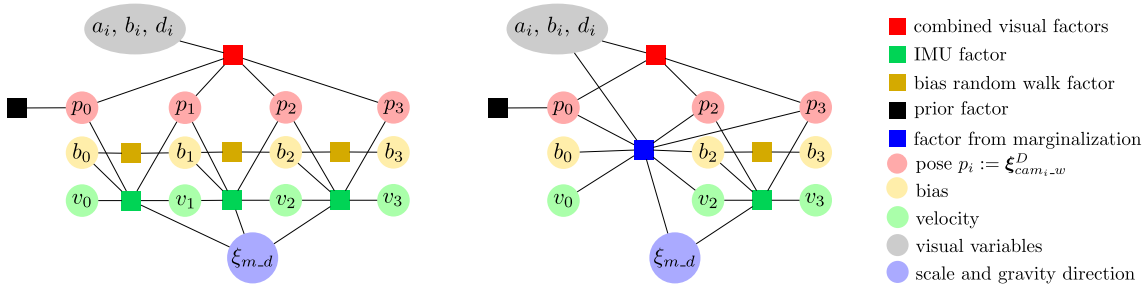
Therefore we propose a novel strategy for handling this issue. We explicitly include scale (and gravity direction) as a parameter in our visual-inertial system and jointly optimize them together with the other values such as poses and geometry. This means that we can initialize with an arbitrary scale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

- We use the same visual initializer as [32] 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.

6.3.5 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 6.2: 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.

6.3.6 Scale-aware Visual-inertial Optimization

We optimize the poses, IMU-biases and velocities of a fixed number of keyframes. Fig. 6.2(a) 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 6.3.3. 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 [95] have done. Note that in contrast to their method however we allow the marginalization procedure described in section 6.3.6 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

$$\boldsymbol{s}_i := [\left(\boldsymbol{\xi}^D_{cam_i\_w}\right)^T, \boldsymbol{v}_i^T, \boldsymbol{b}_i^T, a_i, b_i, d_i^1, d_i^2, ..., d_i^m]^T \tag{6.5}$$

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 (6.2) and d_i^j are the inverse depths

of the points hosted in this keyframe.

The full state vector is then defined as

$$\boldsymbol{s} = [\boldsymbol{c}^T, \boldsymbol{\xi}_{m\_d}^T, \boldsymbol{s}_1^T, \boldsymbol{s}_2^T, ..., \boldsymbol{s}_n^T]^T \tag{6.6}$$

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 6.3.5. We define the operator $\boldsymbol{s} \boxplus \boldsymbol{s'}$ 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 $\boldsymbol{r}$ we define

$$\mathbf{J} = \left.\frac{d\boldsymbol{r}\left(\boldsymbol{s} \boxplus \boldsymbol{\epsilon}\right)}{d\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}, \mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \boldsymbol{b} = -\mathbf{J}^T\mathbf{W}\boldsymbol{r} \tag{6.7}$$

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

$$\mathbf{H} = \mathbf{H}_{\text{photo}} + \mathbf{H}_{\text{imu}} \text{ and } \boldsymbol{b} = \boldsymbol{b}_{\text{photo}} + \boldsymbol{b}_{\text{imu}} \tag{6.8}$$

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.

$$\boldsymbol{s}_i' := [\boldsymbol{\xi}_{w\_imu_i}^M, \boldsymbol{v}_i, \boldsymbol{b}_i]^T \text{ and } \boldsymbol{s'} = \left[\boldsymbol{s}_1'^T, \boldsymbol{s}_2'^T, ..., \boldsymbol{s}_n'^T\right]^T \tag{6.9}$$

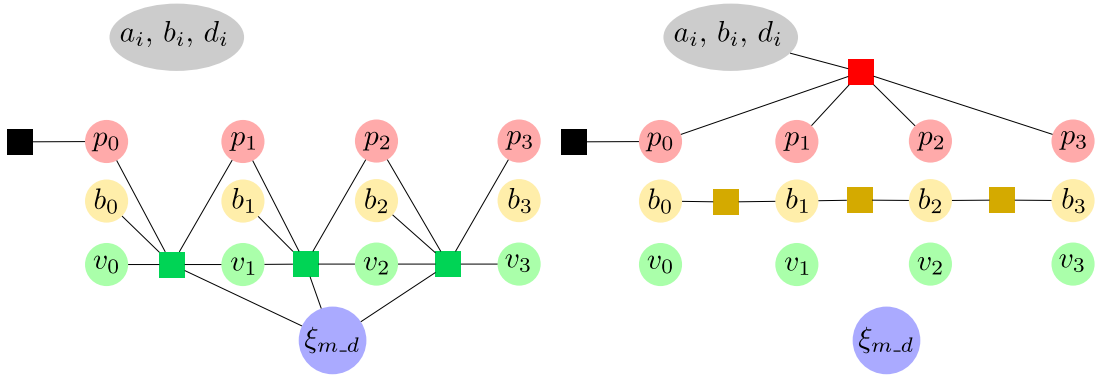
The inertial residuals lead to

$$\mathbf{H}_{\text{imu}}' = \mathbf{J}_{\text{imu}}'^T\mathbf{W}_{\text{imu}}\mathbf{J}_{\text{imu}}' \text{ and } \boldsymbol{b}_{\text{imu}}' = -\mathbf{J}_{\text{imu}}'^T\mathbf{W}_{\text{imu}}\boldsymbol{r}_{\text{imu}} \tag{6.10}$$

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 (6.6). As the two definitions mainly differ in their representation of the poses we can compute $\mathbf{J}_{\text{rel}}$ such that

$$\mathbf{H}_{\text{imu}} = \mathbf{J}_{\text{rel}}^T \cdot \mathbf{H}_{\text{imu}}' \cdot \mathbf{J}_{\text{rel}} \text{ and } \boldsymbol{b}_{\text{imu}} = \mathbf{J}_{\text{rel}}^T \cdot \boldsymbol{b}_{\text{imu}}' \tag{6.11}$$

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 6.3: Partitioning of the factor graph from Fig. 6.2(a) 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. 6.2(b) shows how marginalization changes the factor graph.

The marginalization of the visual factors is handled as in [32] 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 [32, eq. (16), (17) and (18)]. As the factor resulting from marginalization requires the linearization point of all connected variables to remain fixed we apply [32, 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 [32] 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 6.3.6 has two purposes: reduce the computation complexity of the optimization by removing old states and

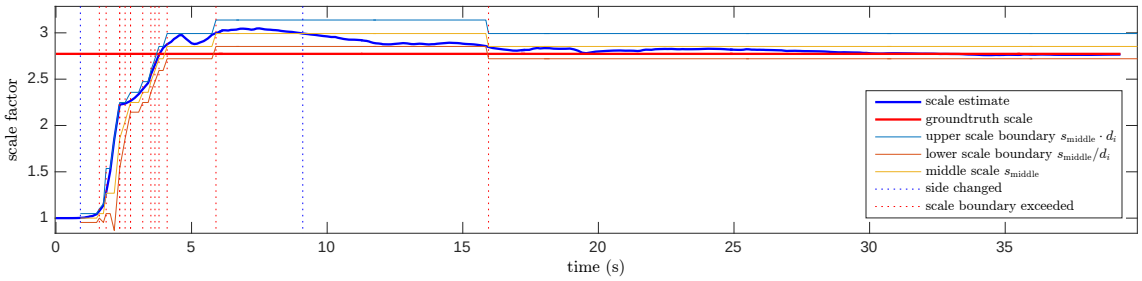


Figure 6.4: The scale estimation running on the V1_03_difficult sequence from the EuRoC dataset. We show the current scale estimate (bold blue), the groundtruth scale (bold red) and the current scale interval (light lines). The vertical dotted lines denote when the side changes (blue) and when the boundary 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.

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 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

$$G_{\text{full}} = G_{\text{metric}} \cup G_{\text{visual}} \tag{6.12}$$

Fig. 6.3 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

$$G_{ba} = G_{\text{metric}} \cup G_{\text{visual}} \cup M_{\text{curr}} \tag{6.13}$$

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

$$s_i := \text{scale estimate at the time, } i \text{ was marginalized} \tag{6.14}$$

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.

$$\forall i \in M_{\text{curr}} : s_i \in [s_{\text{middle}}/d_i, s_{\text{middle}} \cdot d_i] \tag{6.15}$$

$$\forall i \in M_{\text{half}} : s_i \in \begin{cases} [s_{\text{middle}}, s_{\text{middle}} \cdot d_i], & \text{if } s_{\text{curr}} > s_{\text{middle}} \\ [s_{\text{middle}}/d_i, s_{\text{middle}}], & \text{otherwise} \end{cases} \tag{6.16}$$

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 (6.15) and (6.16) we apply Algorithm **??** 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. 6.4 shows how the system works in practice.

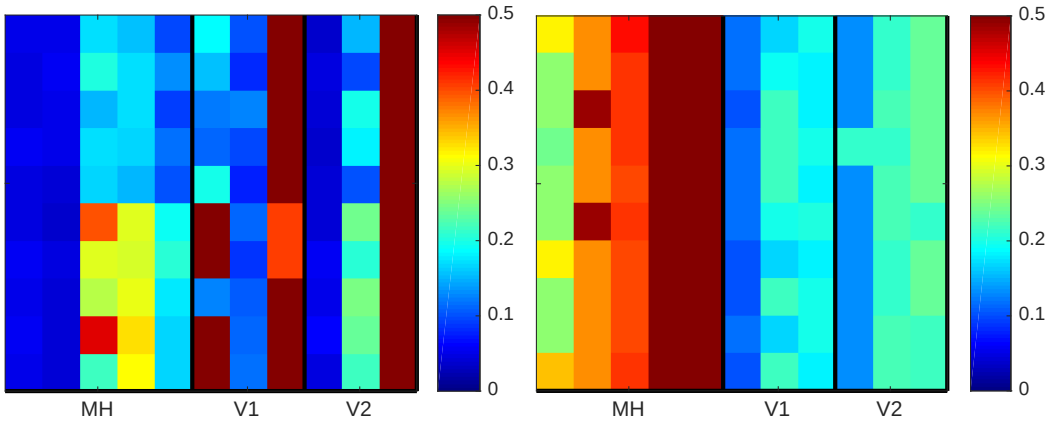
An important part of this strategy is the choice of d_i. It should be small, in order to keep the system consistent, but not too small so that M_{curr} always contains enough inertial factors. Therefore we chose to dynamically adjust the parameter as follows. At all time steps i we calculate

$$d_i = \min \{d_{\min}^j \mid j \in \mathbb{N} \setminus \{0\}, \frac{s_i}{s_{i-1}} < d_i\} \tag{6.17}$$

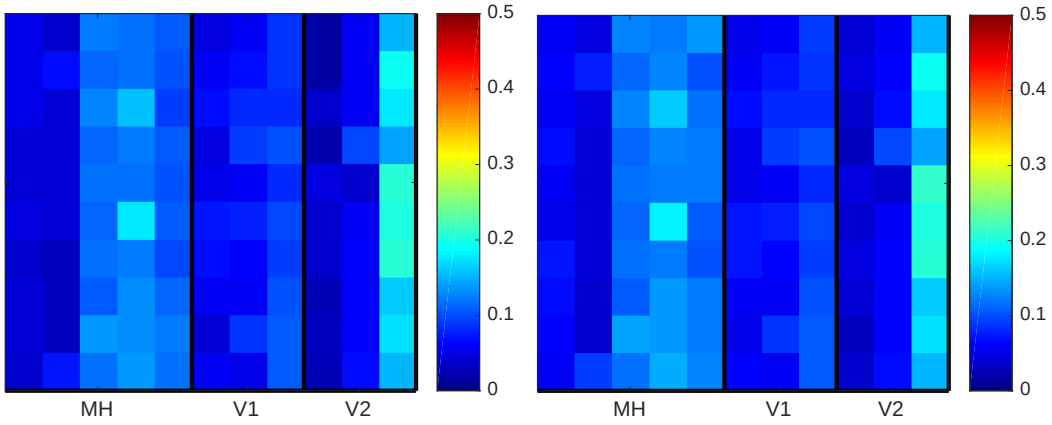
This ensures that it cannot happen that the M_{half} gets reset to M_{visual} at the same time that M_{curr} is exchanged with M_{half}. Therefore it prevents situations where M_{curr} contains no inertial factors at all, making the scale estimation more reliable. In our experiments we chose $d_{\min}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

6.3.7 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 6.3.6. We



(a) DSO [32] realtime gt-scaled (b) ROVIO realtime [17]



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

Figure 6.5: rmse for different methods run 10 times (lines) on each sequence (columns) of the EuRoC dataset.

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.

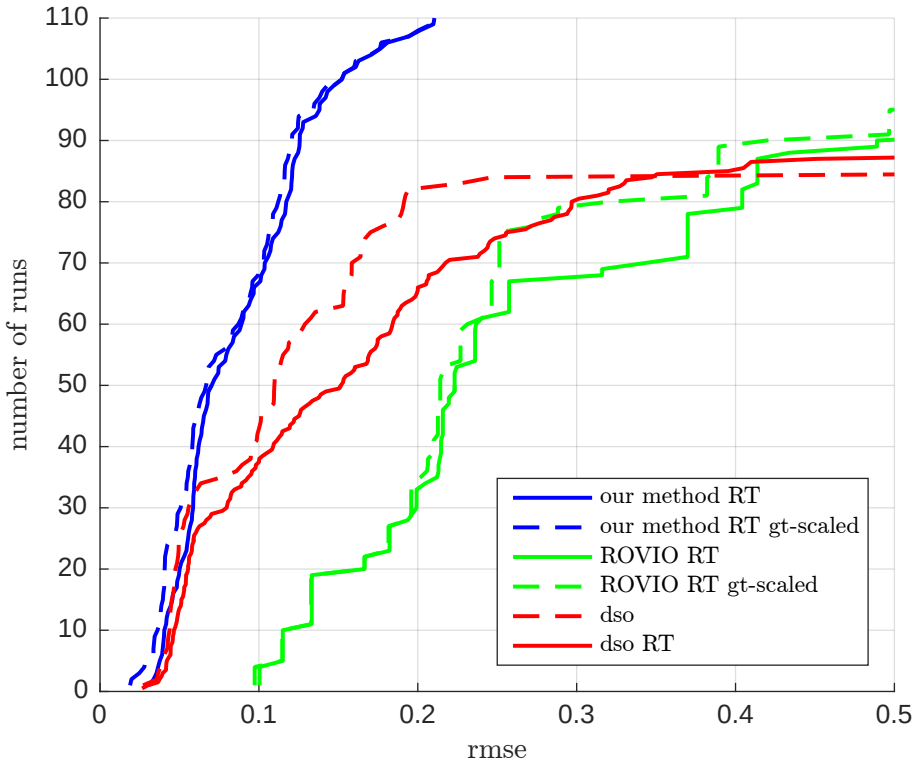


Figure 6.6: Cumulative error plot on the EuRoC-dataset (RT means realtime). This experiment demonstrates that the additional IMU not only provides a reliable scale estimate, but that it also significantly increases accuracy and robustness.

6.4 Results

We evaluate our approach on the publicly available EuRoC dataset [20]. The performance is compared to [32], [17], [94], [8], [73] and [63]. We also provide supplementary material with more evaluation and a video at vision.in.tum.de/vi-dso.

6.4.1 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 [32] and ROVIO [17]. 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

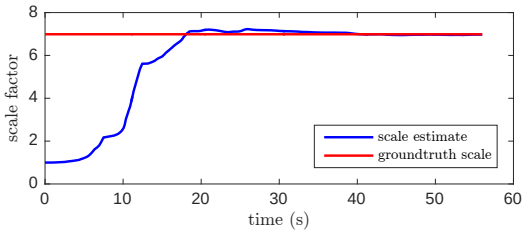


Figure 6.7: Scale estimate for MH_04_difficult (median result of 10 runs in terms of tracking accuracy). Note how the estimated scale converges to the correct value despite being initialized far from the optimum.

challenging for our IMU initialization. Fig. 6.5 shows the root mean square error (rmse) for every run and Fig. 6.6 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 **??** 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. 6.5(c). 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 [73] and [63] were taken from [63]. The results for [94] (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 [73] and [63] 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 [94] and [8] on the V1_0*-sequences of EuRoC (Fig. 6.8). While our method cannot beat the SLAM system and the stereo method on the easy sequence we outperform [8] and are as good as [94] on

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 [73], 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 [73], 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 [63], 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 [63], stereo	RMSE	0.11	0.09	0.19	0.27	0.23	0.04	0.05	0.11	0.10	0.18	X

Table 6.1: 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.

the medium sequence. On the hard sequence we outperform both of the contenders even though we neither use stereo nor loop-closures.

6.4.2 Evaluation of the Initialization

There are only few methods we can compare our initialization to. Some approaches like [85] have not been tested on real data. While [60] 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 [94] as both of the methods work on the challenging EuRoC-dataset and have to estimate the scale, gravity direction, bias, and velocity.

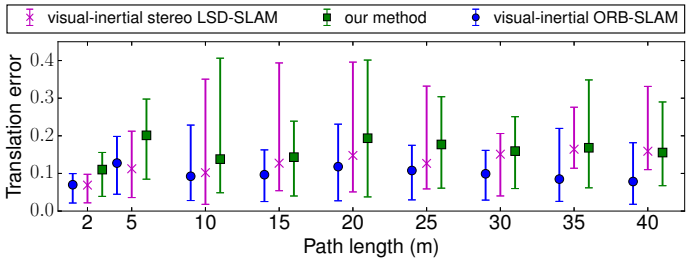
In comparison to [94] our estimated scale is better overall (Table **??**). On most sequences our method provides a better scale, and our average scale error (0.7% compared to 1.0%) as well as our maximum scale error (1.2% compared to 3.4%) is lower. In addition our method is more robust as the initialization procedure of [94] fails on V1_03_difficult.

Apart from the numbers we argue that our approach is superior in terms of the general structure. While [94] 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 [94] 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. 6.7 shows the scale estimation for MH_04.

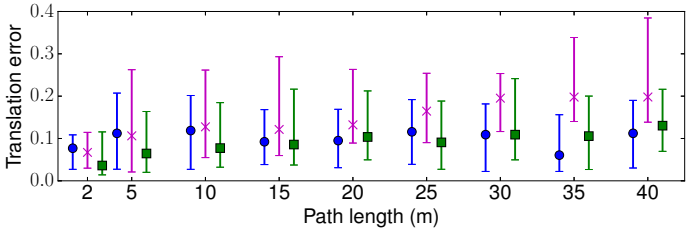
Overall we argue that our initialization procedure exceeds the state of the art and think that the concept of initialization with a very rough scale estimate and jointly estimating it during pose estimation will be a useful concept in the future.

6.5 Conclusion

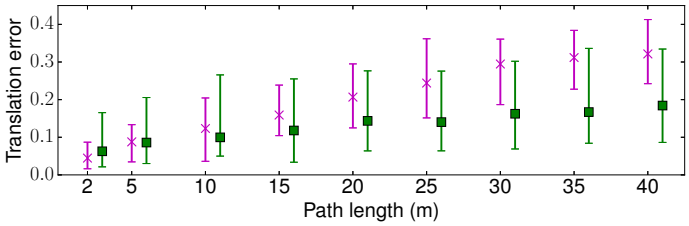
We have presented a novel formulation of direct sparse visual-inertial odometry. We explicitely include scale and gravity direction in our model in order to deal with cases where the scale is not immediately observable. As the initial scale can be very far from the optimum we have proposed a novel technique 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.



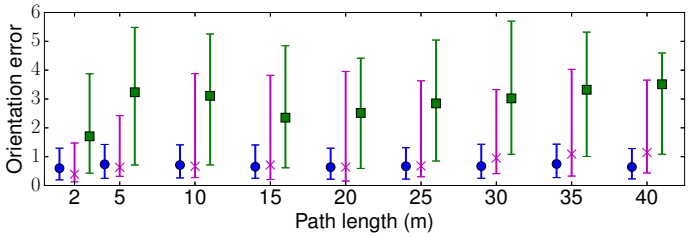
(a) Translation error V1_01_easy



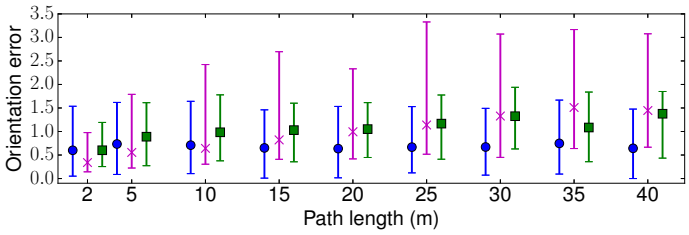
(b) Translation error V1_02_medium



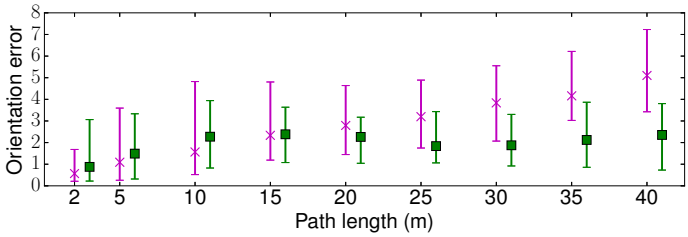
(c) Translation error V1_03_difficult



(d) Orientation error V1_01_easy



(e) Orientation error V1_02_medium



(f) Orientation error V1_03_difficult

Figure 6.8: Relative Pose Error evaluated on three sequences of the EuRoC-dataset for visual-inertial ORB-SLAM [94], visual-inertial stereo LSD-SLAM [8] and our method. Although the proposed VI-DSO does not use loop closuring (like [94]) or stereo (like [8]), VI-DSO is quite competitive in terms of accuracy and robustness. Note that [94] with loop closures is slightly more accurate on average, yet it entirely failed on V1_03_difficult.

Chapter 7

The TUM VI Benchmark for Evaluating Visual-Inertial Odometry

Authors	David Schubert*[1]	schubdav@in.tum.de
	Thore Goll*[1]	gollt@in.tum.de
	Nikolaus Demmel*[1]	demmeln@in.tum.de
	Vladyslav Usenko *[1]	usenko@in.tum.de
	Jörg Stückler [1]	stueckle@in.tum.de
	Daniel Cremers[1]	cremers@tum.de

* These authors contributed equally

[1] Technische Universität München, Munich, Germany

Publication: D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler and D. Cremers. "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robot Systems (IROS)*. Oct. 2018. eprint: http://arxiv.org/abs/1804.06120v1

Contribution		
	Problem definition	*significantly contributed*
	Literature survey	*helped*
	Method development & evaluation	*significantly contributed*
	Implementation	*significantly contributed*
	Experimental evaluation	*helped*
	Preparation of the manuscript	*helped*

Abstract Visual odometry and SLAM methods have a large variety of applications in domains such as augmented reality or robotics. Complementing vision sensors with inertial measurements tremendously improves tracking accuracy and robustness, and thus has spawned large interest in the development of visual-inertial (VI) odometry approaches. In this paper, we propose the TUM VI benchmark, a novel dataset with a diverse set of sequences in different scenes for evaluating VI odometry. It provides camera images with 1024x1024 resolution at 20 Hz, high dynamic range and photometric calibration. An IMU measures accelerations and angular velocities on 3 axes at 200 Hz, while the cameras and IMU sensors are time-synchronized in hardware. For trajectory evaluation, we also provide accurate pose ground truth from a motion capture system at high frequency (120 Hz) at the start and end of the sequences which we accurately aligned with the camera and IMU measurements. The full dataset with raw and calibrated data is publicly available. We also evaluate state-of-the-art VI odometry approaches on our dataset.

7.1 Introduction

Visual odometry and SLAM is a very active field of research with an abundance of applications in fields such as augmented reality or robotics. Variants include monocular ([32, 56]), stereo ([103, 128]) and visual-inertial ([17, 73, 8]) methods. Compared to one camera, adding a second one in a stereo setup provides better robustness and scale-observability. Adding an inertial measurement unit (IMU) helps dealing with untextured environments and rapid motions and makes roll and pitch directly observable. On the other hand, the camera complements the IMU with external referencing to the environment in 6 degrees of freedom.

To compare competing methods, it is necessary to have publicly available data with ground truth. Given the relevance of the topic of visual-inertial odometry, the availability of high-quality datasets is surprisingly small. Compared to single-camera, purely visual datasets, the challenge with a stereo visual-intertial dataset lies in the accurate synchronization of three sensors. A commonly used option for evaluating visual-inertial odometry is the EuRoC MAV dataset [20], but its image resolution and bit depth is not quite state-of-the-art anymore, and the number and variability of scenes is very limited.

For direct methods, which do not align pixel coordinates but image intensities, the assumption that the same 3D point has the same intensity in two different images should be satisfied. It has been shown that providing a photometric calibration that allows to compensate for exposure times, camera response function and lense vignetting is beneficial in this case [32], however it is not a common feature of existing datasets.

Figure 7.1: The **TUM VI benchmark** includes synchronized measurements from an IMU and a stereo camera in several challenging indoor and outdoor sequences. The cameras are equipped with large field-of-view lenses (195°) and provide high dynamic range images (16 bit) at high resolution (1 MP) with linear response function. The figure shows example frames from the dataset.

In this paper, we propose the **TUM VI benchmark**, a novel dataset with a diverse set of sequences in different scenes, with 1024x1024 image resolution at 20 Hz, 16-bit color depth, known exposure times, linear response function and vignette calibration. An IMU provides 3-axis accelerometer and gyro measurements at 200 Hz, which we correct for axis scaling and misalignment, while the cameras and IMU sensors are time-synchronized in hardware. We recorded accurate pose ground truth with a motion capture system at high frequency (120 Hz) which is available at the start and end of the sequences. For accurate alignment of sensor measurements with the ground truth, we calibrated time offsets and relative transforms.

We evaluate state-of-the-art visual-inertial algorithms on our dataset. The full dataset with raw and calibrated data, together with preview videos, is available on:

https://vision.in.tum.de/data/datasets/visual-inertial-dataset

Table 7.1: Comparison of datasets with vision and imu data.

dataset	year	environ.	carrier	cameras	IMUs	time sync	ground truth	stats/props
Kitti Odometry [46]	2013	outdoors	car	1 **stereo** RGB 2x1392x512 @10Hz, 1 stereo gray 2x1392x512 @10Hz	OXTS RT 3003 3-axis acc/gyro @10Hz	sw	OXTS RT 3003 pose @10Hz, acc. <10cm	22 seqs, 39.2 km
Malaga Urban [16]	2014	outdoors	car	1 **stereo** RGB 2x1024x768 @20Hz	3-axis acc/gyro @100Hz	sw	GPS pos @1Hz, low acc	15 sub-seqs, 36.8 km
UMich NCLT [23]	2015	**in-/outdoors**	Segway	6 RGB (omni) 1600x1200 @5Hz	3-axis acc/gyro @100Hz	sw	fused GPS / IMU / laser pose @150Hz, acc≈10cm	27 seqs, 147.3 km
EuRoC MAV [20]	2016	indoors	MAV	1 **stereo** gray 2x752x480 @20Hz	ADIS16488 3-axis acc/gyro @200Hz	**hw**	laser tracker pos @20Hz, **motion capture pose @100Hz**, acc≈1mm	11 seqs, 0.9 km
PennCOSYVIO [107]	2017	**in-/outdoors**	handheld	4 RGB 1920x1080 @30Hz (rolling shutter), 1 **stereo** gray 2x752x480 @20Hz, 1 fish-eye gray 640x480 @30Hz	ADIS16488 3-axis acc/gyro @200Hz, Tango 3-axis acc @128Hz / 3-axis gyro @100Hz	**hw** (stereo gray/ ADIS), sw	fiducial markers pose@30Hz, acc≈15cm	4 seqs, 0.6 km
Zurich Urban MAV [84]	2017	outdoors	MAV	1 RGB 1920x1080 @30Hz (rolling shutter)	3-axis acc/gyro @10Hz	sw	Pix4D visual pose, acc unknown	1 seq, 2 km
Ours (TUM VI)	2018	**in-/outdoors**	handheld	1 **stereo** gray 2x1024x1024 @20Hz	BMI160 3-axis acc/gyro @200Hz	**hw**	**partial motion capture pose @120Hz**, marker pos acc≈1mm (static case)	28 seqs, 20 km, **photometric calibration**

7.2 Related Work

Datasets have in the past greatly fostered the research of visual odometry and SLAM algorithms. In table 7.1 we give an overview over the most relevant datasets that include vision and IMU data.

Visual odometry and SLAM datasets: The TUM RGB-D dataset [124] is focused on the evaluation of RGB-D odometry and SLAM algorithms and has been extensively used by the research community. It provides 47 RGB-D sequences with ground-truth pose trajectories recorded with a motion capture system. It also comes with evaluation tools for measuring drift and SLAM trajectory alignment. For evaluating monocular odometry, recently the TUM MonoVO dataset [1] has been proposed. The dataset contains 50 sequences in indoor and outdoor environments and has been photometrically calibrated for exposure times, lens vignetting and camera response function. Drift can be assessed by comparing the start and end position of the trajectory which coincide for the recordings. We also provide photometric calibration for our dataset, but additionally recorded motion capture ground truth in parts of the trajectories for better pose accuracy assessment. Furthermore, the above datasets do not include time-synchronized IMU measurements with the camera images like our benchmark.

For research on autonomous driving, visual odometry and SLAM datasets have been proposed such as Kitti [46], Malaga Urban dataset [16], or the Robot Oxford car dataset [80]. The Kitti and Malaga Urban datasets also include low-frequency IMU information which is, however, not time-synchronized with the camera images. While Kitti provides a GPS/INS-based ground truth with accuracy below 10 cm, the Malaga Urban dataset only includes a coarse position for reference from a low-cost GPS sensor. Our dataset contains 20 Hz camera images and hardware time-synchronized 3-axis accelerometer and gyro measurements at 200 Hz. Ground-truth poses are recorded at 120 Hz and are accurately time-aligned with the sensor measurements as well.

Visual-inertial odometry and SLAM datasets: Similar to our benchmark, some recent datasets also provide time-synchronized IMU measurements with visual data and have been designed for the evaluation of visual-inertial (VI) odometry and SLAM approaches. The EuRoC MAV dataset [20] includes 11 indoor sequences recorded with a Skybotix stereo VI sensor from a MAV. Accurate ground truth (approx. 1mm) is recorded using a laser tracker or a motion capture system. Compared to our benchmark, the sequences in EuRoC MAV are shorter and have less variety as they only contain recordings in one machine hall and one lab room. Furthermore, EuRoC MAV does not include a photometric calibration which is important to benchmark direct methods. Further datasets for visual-inertial SLAM are the PennCOSYVIO dataset [107] and the Zurich Urban MAV dataset [84]. However, they do not contain photometric calibration and as accurate ground truth or time-

Table 7.2: Overview of sensors in our setup.

Sensor	Type	Rate	Characteristics
Cameras	2 × IDS uEye UI-3241LE-M-GL	20 Hz	global shutter 1024x1024 16-bit gray
IMU	Bosch BMI160	200 Hz	3D accelerometer 3D gyroscope temperature
MoCap	OptiTrack Flex13	120 Hz	6D Pose infrared cameras
Light sensor	TAOS TSL2561	200 Hz	scalar luminance

synchronization of IMU and camera images like our benchmark (cf. table 7.1).

7.3 Sensor Setup

Our sensor setup consists of two monochrome cameras in a stereo setup and an IMU, see fig. 7.2. The left figure shows a schematic view of all involved coordinate systems. We use the convention that a pose $\mathbf{T}_{BA} \in \mathrm{SE}(3)$ transforms point coordinates $\mathbf{p}_A \in \mathbb{R}^3$ in system A to coordinates in B through $\mathbf{p}_B = \mathbf{T}_{BA}\mathbf{p}_A$. For the coordinate systems, we use the following abbreviations,

I IMU

$\mathbf{C}_0$ camera 0

$\mathbf{C}_1$ camera 1

M IR-reflective markers

G grid of AprilTags

W world frame (reference frame of MoCap system)

The IMU is rigidly connected to the two cameras and several IR-reflective markers which allow for pose tracking of the sensor setup by the motion capture (MoCap) system. For calibrating the camera intrinsics and the extrinsics of the sensor setup, we use a grid of AprilTags [104] which has a fixed pose in the MoCap reference (world) system. In the following, we briefly describe the hardware components. An overview is also given in table 7.2.

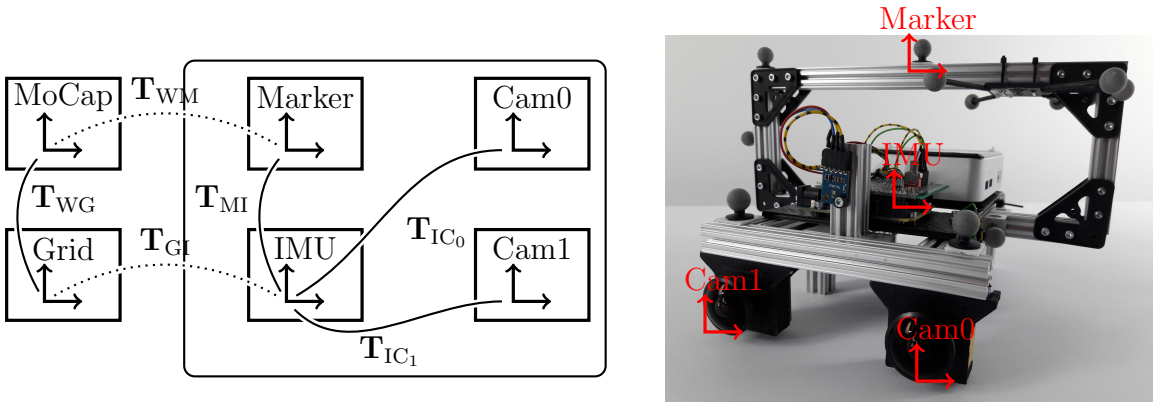


Figure 7.2: Sensor setup. Left: Schematic view of the different coordinate systems. The rounded rectangle contains all components which are rigidly connected with the IMU coordinate system. A dotted line indicates a temporally changing relative pose when moving the sensor. Right: Photo of the sensor setup. It contains two cameras in a stereo setup, a microcontroller board with integrated IMU, a luminance sensor between the cameras and IR reflective markers.

7.3.1 Camera

We use two uEye UI-3241LE-M-GL cameras by IDS. Each has a global shutter CMOS sensor which delivers 1024x1024 monochrome images. The whole intensity range of the sensor can be represented using 16-bit images, so applying a non-linear response function (usually used to increase the precision at a certain intensity range) is not required. The cameras operate at 20 Hz and are triggered synchronously by a Genuino 101 microcontroller.

The cameras are equipped with Lensagon BF2M2020S23 lenses by Lensation. These fisheye lenses have a field of view of 195° (diagonal), though our cameras record a slightly reduced field of view in horizontal and vertical directions due to the sensor size.

7.3.2 Light Sensor

We design our sensor setup to ensure the same exposure time of corresponding images for the two cameras. This way, both camera images have the same brightness for corresponding image points (which otherwise needs to be calibrated or estimated with the visual odometry). Furthermore, this also ensures the same center of the exposure time (which is used as the image timestamp) for two corresponding images and allows us to record accurate per-frame exposure times.

We use a TSL2561 light sensor by TAOS to estimate the required exposure time. The sensor delivers an approximate measurement of the illuminance of the environment. The relation of these measurements and the exposure times which are

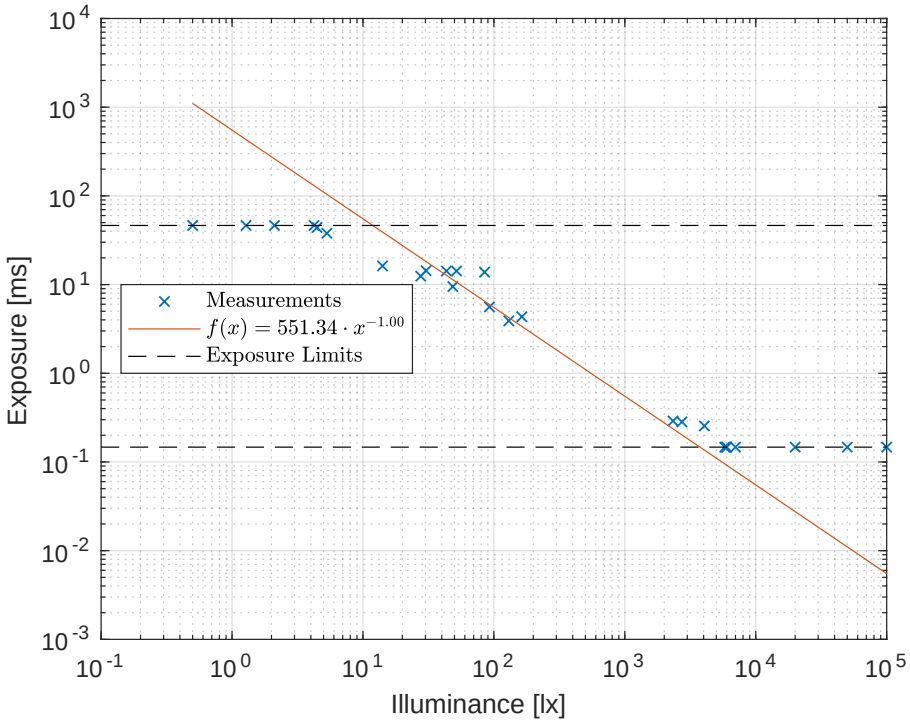


Figure 7.3: Relation of illuminance measurements by our light sensor and corresponding exposure time settings by the camera's auto exposure mode. The dashed lines show the minimum and maximum exposure times possible. The red line shows the least-squares fit (without saturated values) which we use for estimating the next exposure time.

selected by the camera's auto exposure is approximately inversely proportional, as can be seen in fig. 7.3. We find its parameters using a least-squares fit and use it to set the exposure times of both cameras based on the latest illuminance measurement. This assumes that the change in scene brightness between the light measurement and the start of the exposure is negligible. Note that it is not necessary to reproduce the cameras' auto exposure control exactly as long as too dark or too bright images can be avoided. In most cases, the results of our exposure control approach are visually satisfying, but short video segments may be challenging.

7.3.3 IMU

Our sensor setup includes a Bosch BMI160 IMU, which contains 16-bit 3-axis MEMS accelerometer and gyroscope. IMU temperature is recorded, facilitating temperature-dependent noise models. We set its output rate to 200 Hz. The IMU is integrated in the Genuino 101 microcontroller board which triggers the cameras and reads the IMU values. This way, the timestamps of cameras and IMU are well aligned. We estimate the remaining small constant time offset (owing to the read-

out delay of IMU measurements) during the camera-imu extrinsics calibration which yields a value of 5.3 ms for our setup. We estimated this value once and corrected for it in both raw and calibrated datasets.

7.3.4 Motion Capture System

For recording accurate ground-truth poses at a high frame-rate of 120 Hz, we use an OptiTrack motion capture system. It consists of 16 infrared Flex13 cameras which track the IR-reflective markers on the sensor setup. The MoCap system only covers a single room, so we cannot record ground truth for parts of the longer trajectories outside the room. Instead, all sequences start and end in the MoCap room such that our sequences provide ground truth at the beginning and the end.

7.4 Calibration

We include two types of sensor data in our dataset: raw data and calibrated data. The raw data is measured directly by the sensors as described so far, but cannot be used without proper calibration. In the following, we describe which calibrations we apply to the raw data in order to make it usable.

7.4.1 Camera Calibration

Firstly, we calibrate the camera intrinsics and the extrinsics of the stereo setup. We use one of the calib-cam sequences, where we took care to slowly move the cameras in front of the calibration grid to keep motion blur as small as possible.

7.4.2 IMU and Hand-Eye Calibration

We then calibrate the extrinsics between IMU and cameras as well as between IMU and MoCap frame. Concurrently, we estimate the time-synchronization of IMU with MoCap measurements and IMU parameters such as axis alignment, scale differences and biases.

Specifically, we keep the camera intrinsics from the previous calibration fixed and optimize for

- the relative pose between cameras and IMU,
- the time shift between MoCap and IMU time,
- the time shift between camera and IMU time,
- the relative pose between the cameras,
- the relative poses $\mathbf{T}_{\mathrm{MI}}$ and $\mathbf{T}_{\mathrm{WG}}$,

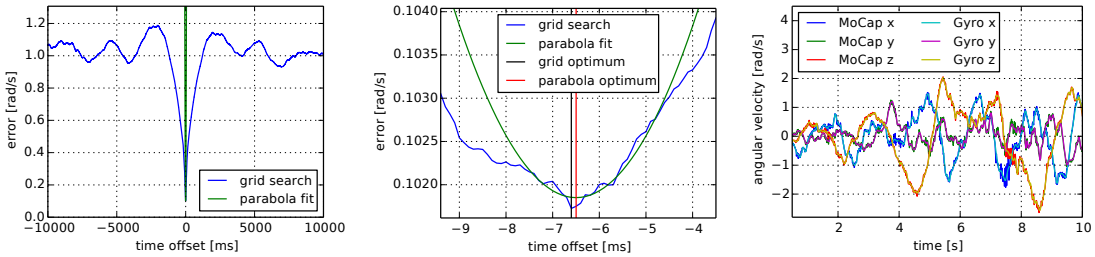


Figure 7.4: Left and middle: Time alignment is performed using grid search. After a coarse initialization it is followed by parabola fitting to find the sub-discretization minimum. Right: Rotational velocities from gyroscope and MoCap after time alignment on the test sequence. MoCap angular velocities are computed using central differences on the orientation.

- coarse initial accelerometer and gyroscope biases $\mathbf{b}_\mathrm{a}$ and $\mathbf{b}_\mathrm{g}$,
- axis scaling and misalignment matrices as in [112] $\mathbf{M}_\mathrm{a}, \mathbf{M}_\mathrm{g} \in \mathbb{R}^{3\times3}$.

The relative poses $\mathbf{T}_\mathrm{MI}$ and $\mathbf{T}_\mathrm{WG}$ are found through hand-eye calibration using a non-linear least squares fitting procedure. Using the relative poses, we convert raw MoCap poses $\mathbf{T}_\mathrm{WM}$ to calibrated ground-truth poses $\mathbf{T}_\mathrm{WI}$ for the IMU.

Additionally, we compensate for the time shift between MoCap and IMU time in the calibrated data. The time offset between MoCap and IMU has to be estimated for each sequence individually. To find the time offset, angular velocities are calculated from the MoCap poses and aligned with the gyroscope measurements. This is done — after a coarse alignment based on measurement arrival time — using a grid search with a stepsize of 100 μs. Then a parabola is fitted around the minimum and the minimum of the parabola is the resulting time offset. The results of this procedure can be seen in fig. 7.4. The ground-truth poses in the calibrated data are always given in IMU time.

We also compensate for axis/scale misalignment and initial biases of the raw accelerations $\mathbf{a}_\mathrm{raw}$ and angular velocities $\boldsymbol{\omega}_\mathrm{raw}$ using

$$\mathbf{a}_\mathrm{calibrated} = \mathbf{M}_\mathrm{a} \cdot \mathbf{a}_\mathrm{raw} - \mathbf{b}_\mathrm{a}\,, \tag{7.1}$$

$$\boldsymbol{\omega}_\mathrm{calibrated} = \mathbf{M}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{raw} - \mathbf{b}_\mathrm{g}\,. \tag{7.2}$$

The matrices $\mathbf{M}_\mathrm{a}, \mathbf{M}_\mathrm{g}$ account for rotational misalignments of gyroscope and accelerometer, axes not being orthogonal or axes not having the same scale. For $\mathbf{M}_\mathrm{g}$, all 9 entries are optimized, whereas $\mathbf{M}_\mathrm{a}$ is chosen to be lower triangular with 6 parameters. The remaining three parameters (rotation) are redundant and have to be fixed in order to obtain a well-constrained system.

In principle, it is not necessary to deduct $\mathbf{b}_\mathrm{a}$ and $\mathbf{b}_\mathrm{g}$, as inertial state estimation algorithms usually estimate a time-varying bias. However, we found that in our

hardware setup there is a large IMU bias that is coarsely reproducible between sensor restarts and therefore approximate precalibration is reasonable. Note that estimating the biases accurately from the sequences is still required for inertial state estimation.

For the calibration step, we use one of the calib-imu sequences which are recorded in front of the calibration grid with motions in all 6 degrees of freedom.

7.4.3 IMU Noise Parameters

For proper probabilistic modeling of IMU measurements in state estimation algorithms and accurate geometric calibration, the intrinsic noise parameters of the IMU are needed. We assume that our IMU measurements (accelerations or angular velocities) are perturbed by white noise with standard deviation σ_{w} and a bias that is slowly changing according to a random walk, which is an integration of white noise with standard deviation σ_{b}. To estimate these quantities, we analyse their Allan deviation $\sigma_{\mathrm{Allan}}(\tau)$ as a function of integration time τ. For a resting IMU with only white noise present, the Allan deviation relates to the white noise standard deviation as

$$\sigma_{\mathrm{Allan}}(\tau) = \frac{\sigma_{\mathrm{w}}}{\sqrt{\tau}}\,, \tag{7.3}$$

so the numerical value of the parameter σ_{w} can be found at $\tau = 1\,\mathrm{s}$. If the measurement is only perturbed by the bias, the relation is

$$\sigma_{\mathrm{Allan}}(\tau) = \sigma_{\mathrm{b}}\sqrt{\frac{\tau}{3}}\,, \tag{7.4}$$

which means the parameter can be found at $\tau = 3\,\mathrm{s}$. The relations between Allan deviation and integration time in Eqs. 7.3 and 7.4 can be found in [53]. White noise and bias dominate the Allan variance in different ranges of τ. Thus, in the log-log plot of $\sigma_{\mathrm{Allan}}(\tau)$ in fig. 7.5, a straight line with slope $-\frac{1}{2}$ has been fitted to an appropriate range of the data to determine σ_{w}, and a straight line with slope $\frac{1}{2}$ has been fitted to another range to determine σ_{b}.

7.4.4 Photometric Calibration

To enable good intensity matching for direct methods, we also provide vignette calibration. For this, we use the calibration code provided by the TUM MonoVO dataset[1] [1]. The image formation model is given by

$$I(\mathbf{x}) = G\left(tV(\mathbf{x})B(\mathbf{x})\right)\,. \tag{7.5}$$

[1] `https://github.com/tum-vision/mono_dataset_code`

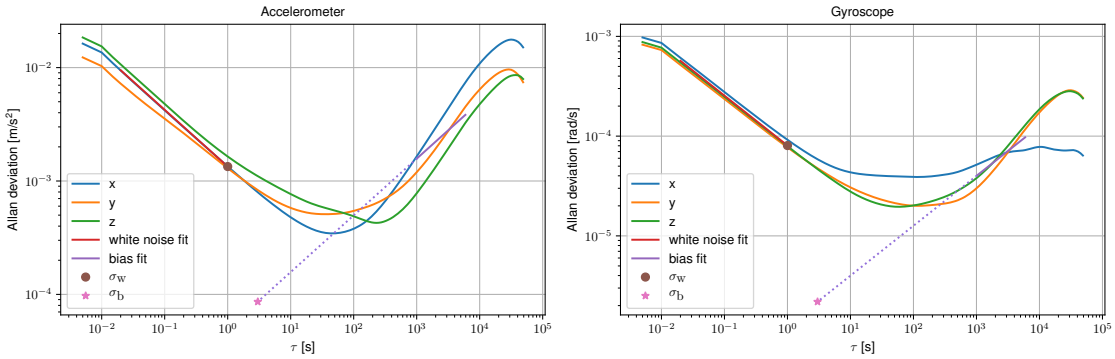


Figure 7.5: Allan deviation of both accelerometer (left) and gyroscope (right). For the fit with slope $-1/2$ we averaged over all three dimensions and took the range $0.02 \leq \tau \leq 1$ into account. For the fit with slope $1/2$, the same averaging was done for the accelerometer, but for the gyroscope we only averaged the y-coordinate and the z-coordinate. The fit region is $1000 \leq \tau \leq 6000$. The assumed slope of $1/2$ does not fit perfectly, which might be due to unmodeled effects such as temperature dependence. The numerical values of noise densities σ_w can be found at an integration time of $\tau = 1s$ on the straight line with slope $-1/2$, while bias parameters σ_b are identified as the value on the straight line with slope $1/2$ at an integration time of $\tau = 3s$. This results in $\sigma_\mathrm{w} = 1.4 \times 10^{-3}\,\mathrm{m/s^2}/\sqrt{\mathrm{Hz}}$, $\sigma_\mathrm{b} = 8.6 \times 10^{-5}\,\mathrm{m/s^3}/\sqrt{\mathrm{Hz}}$ for the accelerometer and $\sigma_\mathrm{w} = 8.0 \times 10^{-5}\,\mathrm{rad/s}/\sqrt{\mathrm{Hz}}$, $\sigma_\mathrm{b} = 2.2 \times 10^{-6}\,\mathrm{rad/s^2}/\sqrt{\mathrm{Hz}}$ for the gyroscope. The white noise parameters are similar to typical values provided by the manufacturer, $\sigma_\mathrm{w} = 1.8 \times 10^{-3}\,\mathrm{m/s^2}/\sqrt{\mathrm{Hz}}$ (accelerometer) and $\sigma_\mathrm{w} = 1.2 \times 10^{-4}\,\mathrm{rad/s}/\sqrt{\mathrm{Hz}}$ (gyroscope).

Table 7.3: RMSE RPE OF THE EVALUATED METHODS ON 1 SECOND SEGMENTS

Sequence	OKVIS	ROVIO	VINS
room1	**0.013m** / **0.43°**	0.029m / 0.53°	0.015m / 0.44°
room2	**0.015m** / **0.62°**	0.030m / 0.67°	0.017m / 0.63°
room3	**0.012m** / **0.63°**	0.027m / 0.66°	0.023m / 0.63°
room4	**0.012m** / 0.57°	0.022m / 0.61°	0.015m / **0.41°**
room5	**0.012m** / **0.47°**	0.031m / 0.60°	0.026m / 0.47°
room6	**0.012m** / 0.49°	0.019m / 0.50°	0.014m / **0.44°**

This means for an image point $\mathbf{x}$, light with intensity $B(\mathbf{x})$ is attenuated by a vignetting factor $V(\mathbf{x}) \in [0, 1]$, then is integrated during the exposure time t, and finally is converted by a response function G into the irradiance value $I(\mathbf{x})$. In our case, we assume G linear, so the model simplifies to $I(\mathbf{x}) \propto tV(\mathbf{x})B(\mathbf{x})$. The given code requires images of a plane with a small calibration tag, taken from different viewpoints. It then alternatingly optimizes the texture of the wall (up to a constant factor) and a non-parametric vignette function. The result is a PNG image representing vignette values between 0 and 1 for each pixel.

7.5 Dataset

7.5.1 Sequences

Besides evaluation sequences, we also make our calibration data accessible such that users can perform their own calibration, even though we provide calibrated data and our calibration results. The sequences can be divided into the following categories.

- **calib-cam:** for calibration of camera intrinsics and stereo extrinsics. A grid of AprilTags has been recorded at low frame rate with changing viewpoints and small camera motion.
- **calib-imu:** for cam-imu calibration to find the relative pose between cameras and IMU. Includes rapid motions in front of the April grid exciting all 6 degrees of freedom. A small exposure has been chosen to avoid motion blur.
- **calib-vignette:** for vignette calibration. Features motion in front of a white wall with a calibration tag in the middle.
- **imu-static:** only IMU data to estimate noise and random walk parameters (111 hours standing still).
- **room:** sequences completely inside the MoCap room such that the full trajectory is covered by the ground truth.

- **corridor:** sequences with camera motion along a corridor and to and from offices
- **magistrale:** sequences featuring a walk around the central hall in a university building
- **outdoors:** sequences of a larger walk outside on a university campus
- **slides:** sequences of a walk in the central hall of a university building including a small part sliding in a closed tube with no visual features.

7.5.2 Format

ROS Bag Files

For each sequence, we provide three different ROS bag files, one raw bag and two calibrated ones. Raw bags contain the data as it has been recorded, i.e. before hand-eye, time shift or IMU calibration. They include the following topics.

`/cam0/image_raw`
`/cam1/image_raw`
`/imu0`
`/vrpn_client/raw_transform`

The first two contain the images of the cameras. Most fields in the messages are self-explanatory and follow standard conventions, but note that `frame_id` provides the exposure time in nanoseconds. In the IMU topic, we do not give the orientation, but we use the second entry of `orientation_covariance` to provide the temperature of the IMU in degree Celsius. The last topic contains the raw MoCap poses $\mathbf{T}_{\mathrm{WM}}$. For each pose there is a timestamp in MoCap time, a translation vector and a rotation quaternion.

Calibrated bags contain the same topics as raw bags but with calibrated data. The differences are:

- MoCap poses have been aligned with the IMU frame (through hand-eye calibration, $\mathbf{T}_{\mathrm{WI}}$),
- outlier MoCap poses have been removed with a median filter on positions,
- timestamps of the MoCap poses have been synchronized with the IMU time using the time shift calibration,
- IMU data has been processed according to eqs. (7.1) and (7.2).

We provide two kinds of calibrated bags: one with full resolution and one with quarter resolution (half resolution for each dimension). The downsampled version facilitates usage for users with storage or bandwidth limitations.

Calibration Files

We also provide geometric calibration files which have been obtained from the processed calibration bags using the Kalibr toolbox[2] [43]. They include intrinsic camera parameters for different models and the relative poses between cameras and IMU. Additionally, the vignette calibration result is given for each camera in PNG format as described in section 7.4.4.

7.6 Evaluation

7.6.1 Evaluation Metric

To evaluate the performance of tracking algorithms on the dataset, we use different evaluation metrics. The *absolute trajectory error* is used, which is the root mean squared difference of ground-truth 3D positions $\hat{\mathbf{p}}_i$ and the corresponding tracked positions $\mathbf{p}_i$, aligned with an optimal SE(3) pose $\mathbf{T}$,

$$r_{\text{ate}} = \min_{\mathbf{T}\in\text{SE}(3)} \sqrt{\frac{1}{|I_{\text{gt}}|} \sum_{i\in I_{\text{gt}}} \left\|\mathbf{T}\mathbf{p}_i - \hat{\mathbf{p}}_i\right\|^2}\,. \tag{7.6}$$

All tracked poses where ground truth is available are used, which corresponds to indices I_{gt}. For most sequences, this is the case at the start and at the end, but for some sequences, there is ground truth throughout.

For visual odometry without global optimization, another reasonable quantity is the *relative pose error*. Following [124], it is defined as

$$r_{\text{rpe}} = \sqrt{\frac{1}{|I_{\text{gt},\Delta}|} \sum_{i\in I_{\text{gt},\Delta}} \left\|\text{trans}(\mathbf{E}_i)\right\|^2}\,, \tag{7.7}$$

$$\mathbf{E}_i = \left(\hat{\mathbf{T}}_i^{-1}\hat{\mathbf{T}}_{i+\Delta}\right)^{-1} \left(\mathbf{T}_i^{-1}\mathbf{T}_{i+\Delta}\right)\,, \tag{7.8}$$

where $\text{trans}(\cdot)$ takes the 3D translational component of a pose. This error measures how accurate pose changes are in a small time interval Δ. The set of frame indices $I_{\text{gt},\Delta}$ is the same as I_{gt}, but we have to take out Δ poses at the end of each tracked segment.

7.6.2 Results

To verify that the dataset is suitable for benchmarking visual-inertial odometry systems, we provide the results of several state-of-the-art methods that have open-source implementations. Unless specified otherwise, the methods are used with

[2] `https://github.com/ethz-asl/kalibr`

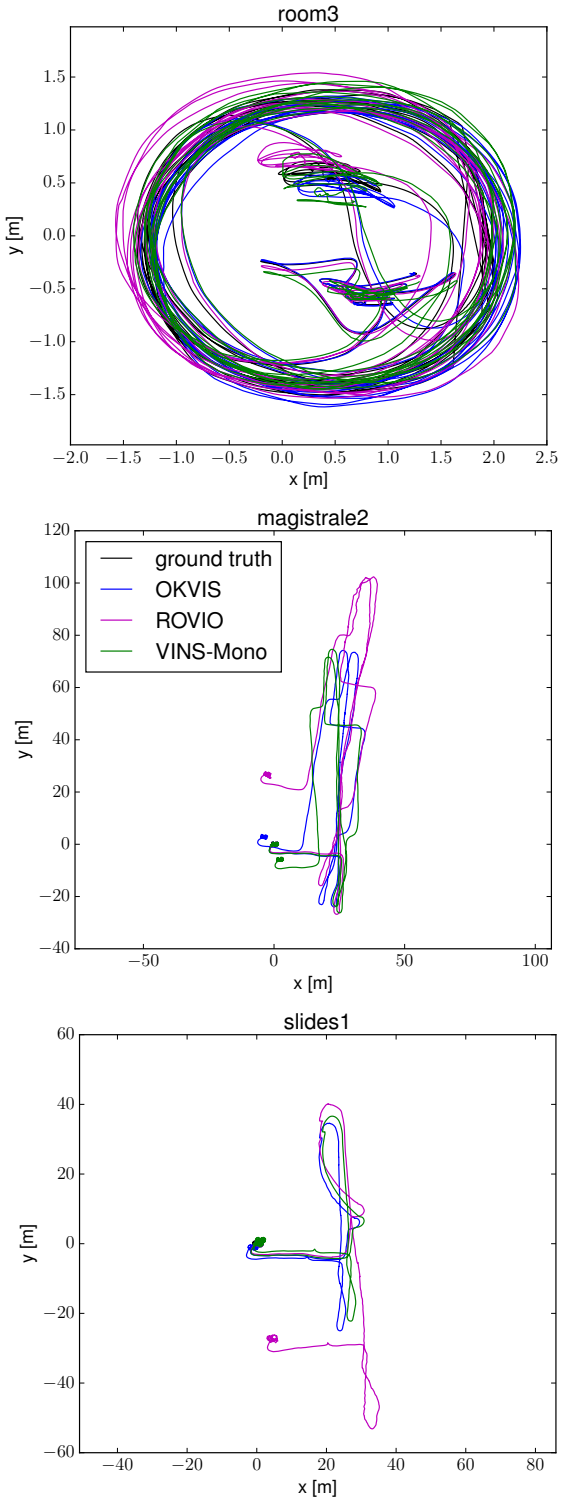


Figure 7.6: Results of evaluated methods for room3, magistrale2 and slides1 sequences from our dataset. The ground truth is shown in black for the segments of the trajectory where it is available. The presented results are obtained with synchronous processing, without enforcing real-time and otherwise default parameters (except VINS-Mono for which non-realtime version is not implemented). Noise parameters are set to inflated values from the Allan plots in fig. 7.5 to account for unmodeled noise and vibrations.

default parameters on quarter resolution images (512x512 pixels). We found that most of the algorithms have default parameters tuned to images with VGA resolution, which makes their performance better on sub-sampled datasets, while full resolution data might be useful for future research.

We provide evaluations for ROVIO [17], OKVIS [73] and VINS-Mono [110]. The results are summarised in table 7.3 and table 7.4 and a visualization for some sequences is presented in fig. 7.6. All systems are able to track most of the sequences until the end, surprisingly, even the sequences with complete absence of visual features for some parts of the trajectory (slides). However, sometimes the estimators diverge at some point during the sequence, which results in erratic translation or rapid drift. We call a sequence diverged, if the ATE based on just the end-segment is larger[3] than 2 m, which is indicated by underlines in table 7.4. The ATE values are still informative, as most often divergence happens towards the end (values larger than 1000 m are shown as "X").

OKVIS and VINS-Mono perform mostly well, but struggle for some of the longer outdoor sequences. ROVIO is more prone to drift and diverges on several sequences, which might be explained by it's use of a Kalman filter compared to computationally more demanding non-linear least squares optimization employed by OKVIS and VINS-Mono. VINS-Mono diverges on most of the outdoor sequences, but typically only after the camera returns to the motion capture room and switches from mainly forward motion to fast rotations. This might indicate a drift in accelerometer bias estimates.

The evaluation shows that even the best performing algorithms have significant drift in long (magistrale, outdoors) and visually challenging (slides) sequences. This means that the dataset is challenging enough to be used as a benchmark for further research in visual-inertial odometry algorithms.

7.7 Conclusion

In this paper, we proposed a novel dataset with a diverse set of sequences in different scenes for evaluating visual-inertial odometry. It contains high resolution images with high dynamic range and vignette calibration, hardware synchronized with 3-axis accelerometer and gyro measurements. For evaluation, the dataset contains accurate pose ground truth at high frequency at the start and end of the sequences. We perform hand-eye calibration on calibration sequences and time-offset estimation on all sequences to have ground truth data geometrically and temporally aligned with the IMU. In addition, we provide sequences to calibrate IMU white noise and random walk and vignetting of the camera. The dataset is publicly available with raw and calibrated data.

[3]For all evaluated systems, median values over all sequences for ATE based on just the start-segment are less than 0.1 m and less than 0.5 m for just the end-segment.

Table 7.4: RMSE ATE IN M OF THE EVALUATED METHODS

Sequence	OKVIS	ROVIO	VINS	length [m]
corridor1	**0.33**	0.47	0.63	305
corridor2	**0.47**	0.75	0.95	322
corridor3	**0.57**	0.85	1.56	300
corridor4	0.26	**0.13**	0.25	114
corridor5	**0.39**	2.09	0.77	270
magistrale1	3.49	4.52	**2.19**	918
magistrale2	**2.73**	13.43	3.11	561
magistrale3	1.22	14.80	**0.40**	566
magistrale4	**0.77**	39.73	5.12	688
magistrale5	1.62	3.47	**0.85**	458
magistrale6	3.91	X	**2.29**	771
outdoors1	X	101.95	**74.96**	2656
outdoors2	73.86	**21.67**	133.46	1601
outdoors3	32.38	**26.10**	36.99	1531
outdoors4	19.51	X	**16.46**	928
outdoors5	**13.12**	54.32	130.63	1168
outdoors6	**96.51**	149.14	133.60	2045
outdoors7	**13.61**	49.01	21.90	1748
outdoors8	**16.31**	36.03	83.36	986
room1	**0.06**	0.16	0.07	146
room2	0.11	0.33	**0.07**	142
room3	**0.07**	0.15	0.11	135
room4	**0.03**	0.09	0.04	68
room5	**0.07**	0.12	0.20	131
room6	**0.04**	0.05	0.08	67
slides1	0.86	13.73	**0.68**	289
slides2	2.15	**0.81**	0.84	299
slides3	2.58	4.68	**0.69**	383

We also use our benchmark to evaluate the performance of state-of-the-art monocular and stereo visual-inertial methods. Our results demonstrate several open challenges for such approaches. Hence, our benchmark can be useful for the research community for evaluating visual-inertial odometry approaches in future research.

Chapter 8

Real-Time Trajectory Replanning for MAVs using Uniform B-splines and a 3D Circular Buffer

Authors	Vladyslav Usenko[1]	usenko@in.tum.de
	Lukas von Stumberg [1]	stumberg@in.tum.de
	Andrej Pangercic [1]	pangerca@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1]Technische Universität München, Munich, Germany	

Publication 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: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8202160

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 In this paper, we present a real-time approach to local trajectory replanning for microaerial vehicles (MAVs). Current trajectory generation methods for multicopters achieve high success rates in cluttered environments, but assume that the environment is static and require prior knowledge of the map. In the presented study, we use the results of such planners and extend them with a local replanning algorithm that can handle unmodeled (possibly dynamic) obstacles while keeping the MAV close to the global trajectory. To ensure that the proposed approach is real-time capable, we maintain information about the environment around the MAV in an occupancy grid stored in a three-dimensional circular buffer, which moves together with a drone, and represent the trajectories by using uniform B-splines. This representation ensures that the trajectory is sufficiently smooth and simultaneously allows for efficient optimization.

8.1 Introduction

In recent years, microaerial vehicles (MAVs) have gained popularity in many practical applications such as aerial photography, inspection, surveillance and even delivery of goods. Most commercially available drones assume that the path planned by the user is collision-free or provide only limited obstacle-avoidance capabilities. To ensure safe navigation in the presence of unpredicted obstacles a replanning method that generates a collision-free trajectory is required.

Formulation of the trajectory generation problem largely depends on the application and assumptions about the environment. In the case where an MAV is required to navigate a cluttered environment, possibly an indoor one, we would suggest subdividing the problem into two layers. First, we assume that a map of the environment is available and a trajectory from a specified start point to the goal point is planned in advance.

This task has been a popular research topic in recent years, and several solutions have been proposed by Achtelik et al. [11] and Richter et al. [113]. They used occupancy representation of the environment to check for collisions and searched for the valid path in a visibility graph constructed using sampling based planners. Thereafter, they followed the approach proposed by Mellinger and Kumar [88] to fit polynomial splines through the points of the planned path to generate a smooth feasible trajectory. The best algorithms of this type can compute a trajectory through tens of waypoints in several seconds.

To cope with any unmodeled, possibly dynamic, obstacle a lower planning level is required, which can generate a trajectory that keeps the MAV close to the global path and simultaneously avoids unpredicted obstacles based on an environment representation constructed from the most recent sensor measurements. This replanning level should run in several milliseconds to ensure the safety of MAVs operating at

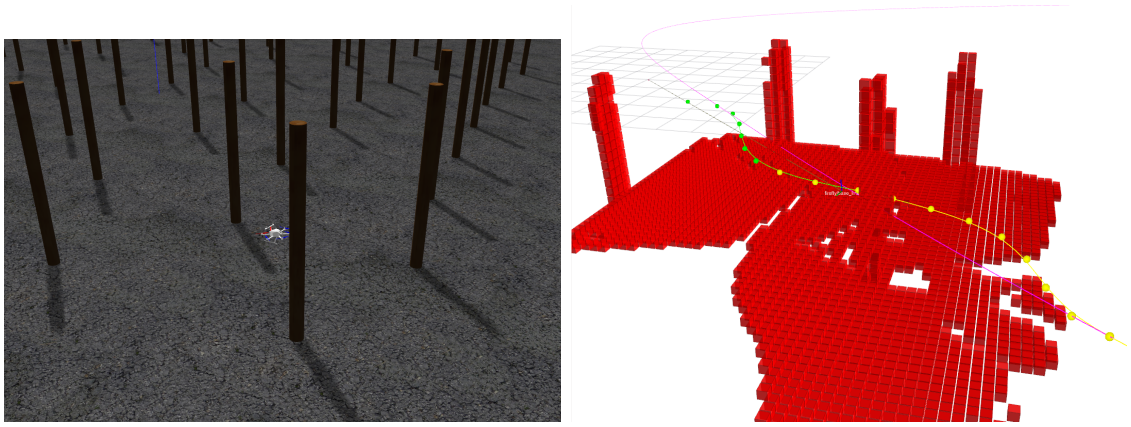


Figure 8.1: Example of local trajectory replanning algorithm running in the simulator. Global trajectory is visualized in purple and the local obstacle map is visualized in red. The local trajectory is represented by a uniform quintic B-spline, and its control points are visualized in yellow for the fixed parts and in green for the parts that can still change due to optimization.

high velocities.

The proposed approach solves a similar problem as that solved by Oleynikova et al. [101], but instead of using polynomial splines for representing the trajectory we propose the use of B-splines and discuss their advantages over polynomial splines for this task. Furthermore, we propose the use of a robocentric, fixed-size three-dimensional (3D) circular buffer to maintain local information about the environment. Even though the proposed method cannot model arbitrarily large occupancy maps, as some octree implementations, faster look-up and measurement insertion operations make it better suited for real-time replanning tasks.

We demonstrate the performance of the system in several simulated and real-world experiments, and provide open-source implementation of the software to community.

The contributions of the present study are as follows:

- Formulation of local trajectory replanning as a B-spline optimization problem and thorough comparison with alternative representations (polynomial, discrete).
- High-performance 3D circular buffer implementation for local obstacle mapping and collision checking and comparison with alternative methods.
- System design and evaluation on realistic simulator and real hardware, in addition to performance comparison with existing methods.

In addition to analyzing the results presented in the paper, we encourage the reader to watch the demonstration video and inspect the available code, which can be found at

`https://vision.in.tum.de/research/robotvision/replanning`

8.2 Related Work

In this section, we describe the studies relevant to different aspects of collision-free trajectory generation. First, we discuss existing trajectory generation strategies and their applications to MAV motion planning. Thereafter, we discuss the state-of-the-art approaches for 3D environment mapping.

8.2.1 Trajectory Generation

Trajectory generation strategies can be subdivided into three main approaches: search-based path planning followed by smoothing, optimization-based approaches and motion-primitive-based approaches.

In search-based approaches, first, a non-smooth path is constructed on a graph that represents the environment. This graph can be a fully connected grid as in [31] and [59], or be computed using a sampling-based planner (RRT, PRM) as in [113] and [21]. Thereafter, a smooth trajectory represented by a polynomial, B-spline or discrete set of points is computed to closely follow this path. This class of approaches is currently the most popular choice for large-scale path planning problems in cluttered environments where a map is available a priory.

Optimization-based approaches rely on minimizing a cost function that consists of smoothness and collision terms. The trajectory itself can be represented as a set of discrete points [133] or polynomial segments [101]. The approach presented in the present work falls into this category, but represents a trajectory using uniform B-splines.

Another group of approaches is based on path sampling and motion primitives. Sampling-based approaches were successfully used for challenging tasks such as ball juggling [93], and motion primitives were successfully applied to flight through the forest [105], but the ability of both approaches to find a feasible trajectory depends largely on the selected discretization scheme.

8.2.2 Environment Representation

To be able to plan a collision-free trajectory a representation of the environment that stores information about occupancy is required. The simplest solution that can be used in the 3D case is a voxel grid. In this representation, a volume is subdivided into regular grid of smaller sub-volumes (voxels), where each voxel stores information about its occupancy. The main drawback of this approach is its large memory-footprint, which allows for maping only small fixed-size volumes. The advantage, however, is very fast constant time access to any element.

To deal with the memory limitation, octree-based representations of the environment are used in [52] [122]. They store information in an efficient way by pruning the leaves of the trees that contain the same information, but the access times for each element become logarithmic in the number of nodes, instead of the constant time as in the voxel-based approaches.

Another popular approach to environment mapping is voxel hashing, which was proposed by Nießner et al. [98] and used in [102]. It is mainly used for storing a truncated signed distance function representation of the environment. In this case, only a narrow band of measurements around the surface is inserted and only the memory required for that sub-volume is allocated. However, when full measurements must be inserted or the dense information must be stored the advantages of this approach compared to those of the other approaches are not significant.

8.3 Trajectory Representation using Uniform B-Splines

We use uniform B-spline representation for the trajectory function $p(t)$. Because, as shown in [88] and [11], the trajectory must be continuous up to the forth derivative of position (snap), we use quintic B-splines to ensure the required smoothness of the trajectory.

8.3.1 Uniform B-Splines

The value of a B-spline of degree $k-1$ can be evaluated using the following equation:

$$p(t) = \sum_{i=0}^{n} p_i B_{i,k}(t), \tag{8.1}$$

where $p_i \in \mathbb{R}^n$ are control points at times $t_i, i \in [0, .., n]$ and $B_{i,k}(t)$ are basis functions that can be computed using the De Boor – Cox recursive formula [18] [28]. Uniform B-splines have a fixed time interval Δt between their control points, which simplifies computation of the basis functions.

In the case of quintic uniform B-splines, at time $t \in [t_i, t_{i+1})$ the value of $p(t)$ depends only on six control points, namely $[t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}, t_{i+3}]$. To simplify calculations we transform time to a uniform representation $s(t) = (t - t_0)/\Delta t$, such that the control points transform into $s_i \in [0, .., n]$. We define function $u(t) = s(t) - s_i$ as time elapsed since the start of the segment. Following the matrix representation of the De Boor – Cox formula [109], the value of the function can be evaluated as follows:

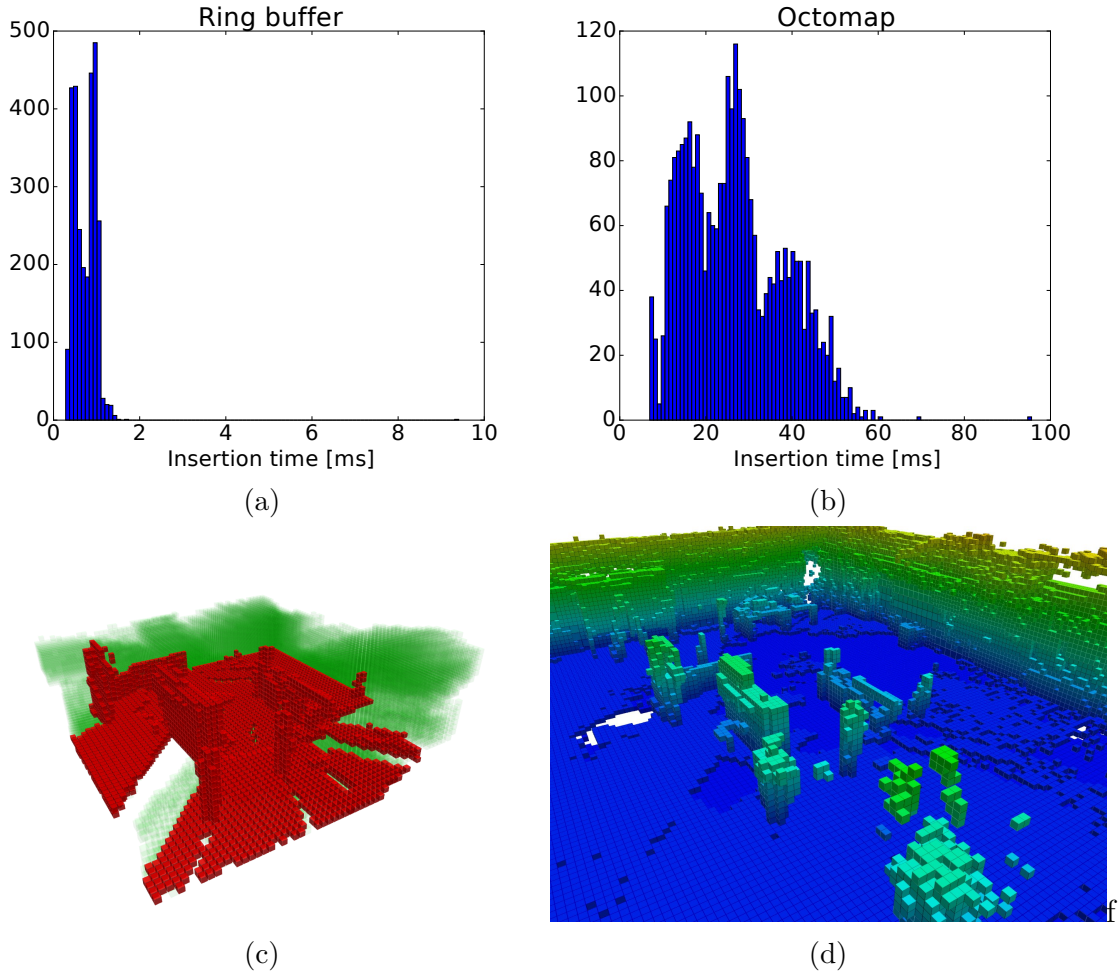


Figure 8.2: Comparison between octomap and circular buffer for occupancy mapping on fr2/pioneer_slam2 sequence of [124]. Being able to map only a local environment around the robot (3 m at voxel resolution of 0.1 m) circular buffer is more than an order of magnitude faster when inserting point cloud measurements from a depth map subsampled to a resolution of 160×120. Subplots ((a)) and ((b)) show the histograms of insertion time, and ((c)) and ((d)) show the qualitative results of the circular buffer (red: occupied, green:free) and the octomap, respectively.

$$p(u(t)) = \begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \\ u^4 \\ u^5 \end{pmatrix}^T M_6 \begin{pmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{pmatrix}, \tag{8.2}$$

$$M_6 = \frac{1}{5!} \begin{pmatrix} 1 & 26 & 66 & 26 & 1 & 0 \\ -5 & -50 & 0 & 50 & 5 & 0 \\ 10 & 20 & -60 & 20 & 10 & 0 \\ -10 & 20 & 0 & -20 & 10 & 0 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{pmatrix}. \tag{8.3}$$

Given this formula, we can compute derivatives with respect to time (velocity, acceleration) as follows:

$$p'(u(t)) = \frac{1}{\Delta t} \begin{pmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \\ 4u^3 \\ 5u^4 \end{pmatrix}^T M_6 \begin{pmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{pmatrix}, \tag{8.4}$$

$$p''(u(t)) = \frac{1}{\Delta t^2} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 6u \\ 12u^2 \\ 20u^3 \end{pmatrix}^T M_6 \begin{pmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{pmatrix}. \tag{8.5}$$

The computation of other time derivatives and derivatives with respect to control points is also straightforward.

The integral over squared time derivatives can be computed in the closed form.

For example, the integral over squared acceleration can be computed as follows:

$$E_q = \int_{t_i}^{t_{i+1}} p''(u(t))^2 dt \tag{8.6}$$

$$= \begin{pmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{pmatrix}^T M_6^T Q M_6 \begin{pmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{pmatrix}, \tag{8.7}$$

where

$$Q = \frac{1}{\Delta t^3} \int_0^1 \begin{pmatrix} 0 \\ 0 \\ 2 \\ 6u \\ 12u^2 \\ 20u^3 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 6u \\ 12u^2 \\ 20u^3 \end{pmatrix}^T du \tag{8.8}$$

$$= \frac{1}{\Delta t^3} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 12 & 16 & 20 \\ 0 & 0 & 12 & 24 & 36 & 48 \\ 0 & 0 & 16 & 36 & 57.6 & 80 \\ 0 & 0 & 20 & 48 & 80 & 114.286 \end{pmatrix}. \tag{8.9}$$

Please note that matrix Q is constant in the case of uniform B-splines. Therefore, it can be computed in advance for determining the integral over any squared derivative (see [113] for details).

8.3.2 Comparison with Polynomial Trajectory Representation

In this subsection, we discuss the advantages and disadvantages of B-spline trajectory representation compared to polynomial-splines-based representation [113] [101].

To obtain a trajectory that is continuous up to the forth derivative of position, we need to use B-splines of degree five or greater and polynomial splines of at least degree nine (we need to set five boundary constraints on each endpoint of the segment). Furthermore, for polynomial splines we must explicitly include boundary constraints into optimization, while the use of B-splines guarantees the generation of a smooth trajectory for an arbitrary set of control points. Another useful property of B-splines is the locality of trajectory changes due to changes in the control points,

which means that a change in one control point affects only a few segments in the entire trajectory. All these properties result in faster optimization because there are fewer variables to optimize and fewer constraints.

Evaluation of position at a given time, derivatives with respect to time (velocity, acceleration, jerk, snap), and integrals over squared time derivatives are similar for both cases because closed-form solutions are available for both cases.

The drawback of B-splines, however, is the fact that the trajectory does not pass through the control points. This makes it difficult to enforce boundary constraints. The only constraint we can enforce is a static one (all time derivatives are zero), which can be achieved by inserting the same control point $k + 1$ times, where k is the degree of the B-spline. If we need to set an endpoint of the trajectory to have a non-zero time derivative, an iterative optimization algorithm must be used.

These properties make polynomial splines more suitable for the cases where the control points come from planning algorithms (RRT, PRM), which means that the trajectory must pass through them, else the path cannot be guaranteed to be collision-free. For local replanning, which must account for unmodeled obstacles, this property is not very important; thus, the use of B-spline trajectory representation is a better option.

8.4 Local Environment Map using 3D Circular Buffer

To avoid obstacles during flight, we need to maintain an occupancy model of the environment. On one hand, the model should rely on the most recent sensor measurements, and on the other hand it should store some information over time because the field of view of the sensors mounted on the MAV is usually limited.

We argue that for local trajectory replanning a simple solution with a robocentric 3D circular buffer is beneficial. In what follows, we discuss details pertaining to implementation and advantages from the application viewpoint.

8.4.1 Addressing

To enable addressing we discretize the volume into voxels of size r. This gives us a mapping from point p in 3D space to an integer valued index x that identifies a particular voxel, and the inverse operation that given an index outputs its center point.

A circular buffer consists of a continuous array of size N and an offset index o that defines the location of the coordinate system of the volume. With this information, we can define the functions to check whether a voxel is in the volume and find its

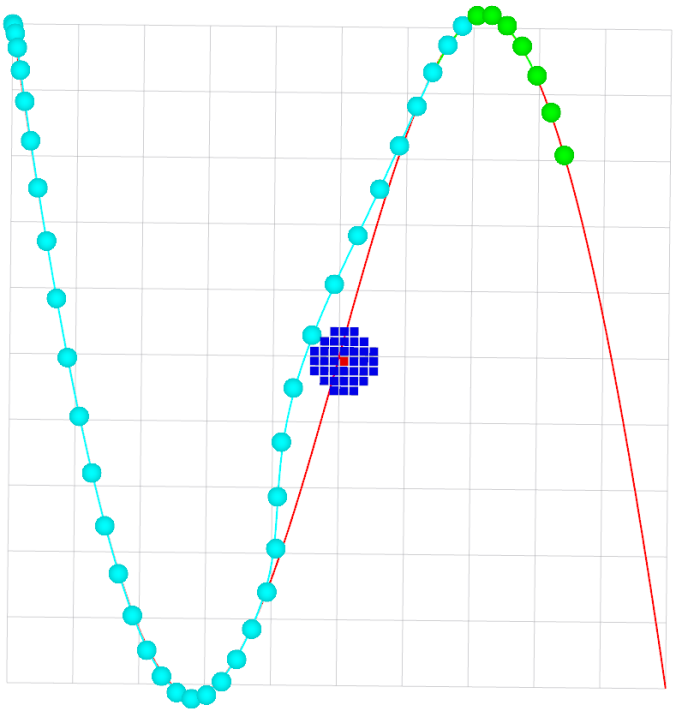
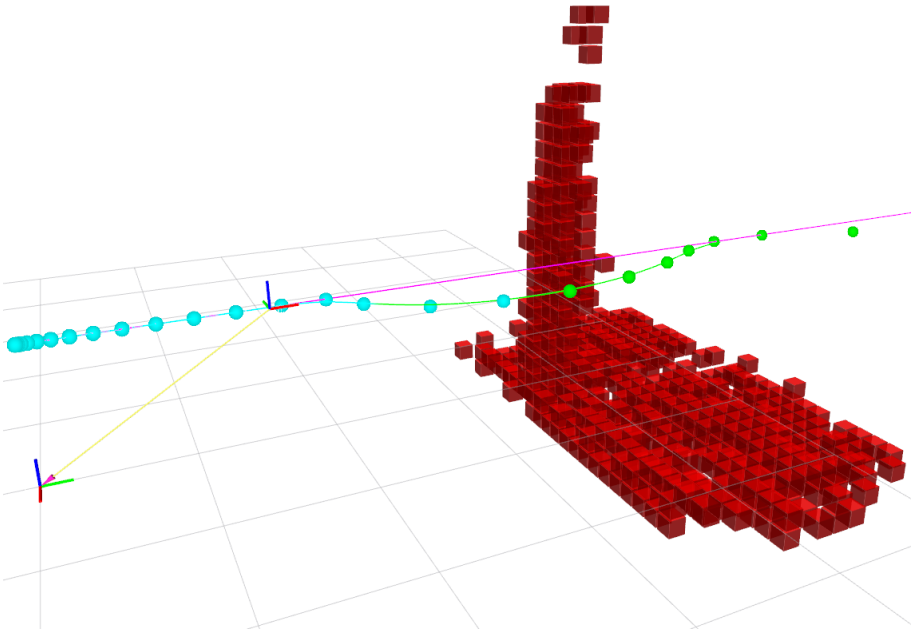


Figure 8.3: Example of online trajectory replanning using proposed optimization objective. The plot shows a global trajectory computed by fitting a polynomial spline through fixed waypoints (red), voxels within 0.5 m of the obstacle (blue), computed B-spline trajectory with fixed (cyan) and still optimized (green) segments and control points. In the areas with no obstacles, the computed trajectory closely follows the global one, while close to an obstacle, the proposed method generates a smooth trajectory that avoids the obstacle, and rejoins the global trajectory.

(a) (b)



(c)

Figure 8.4: Real-world experiment performed outdoors. The drone (AscTec Neo) equipped with RGB-D camera (Intel Realsense R200) is shown in ((a)). In the experiment, the global path is set to a straight line with the goal position 30 m ahead of the drone, and trees act as unmapped obstacles that the drone must avoid. Side view of the scene is shown in ((b)), and visualization with the planned trajectory is shown in ((c)).

address in the stored array:

$$insideVolume(x) = 0 \leq x - o < N, \tag{8.10}$$
$$address(x) = (x - o) \mod N. \tag{8.11}$$

If we restrict the size of the array to $N = 2^p$, we can rewrite these functions to use cheap bitwise operations instead of divisions:

$$insideVolume(x) = !\ ((x - o)\ \&\ (\sim (2^p - 1))), \tag{8.12}$$
$$address(x) = (x - o)\ \&\ (2^p - 1). \tag{8.13}$$

where & is a "bitwise and," $\sim$ is a "bitwise negation," and ! is a "boolean not.".

To ensure that the volume is centered around the camera, we must simply change the offset o and clear the updated part of the volume. This eliminates the need to copy large amounts of data when the robot moves.

8.4.2 Measurement Insertion

We assume that the measurements are performed using range sensors, such as Lidar, RGB-D cameras, and stereo cameras, and can be inserted into the occupancy buffer by using raycast operations.

We use an additional flag buffer to store a set of voxels affected by insertion. First, we iterate over all points in our measurements, and for the points that lie inside the volume, we mark the corresponding voxels as occupied. For the points that lie outside the volume, we compute the closest point inside the volume and mark the corresponding voxels as free rays. Second, we iterate over all marked voxels and perform raycasting toward the sensor origin. We use a 3D variant of *Bresenham's line algorithm* [13] to increase the efficiency of the raycasting operation.

Thereafter, we again iterate over the volume and update the volume elements by using the hit and miss probabilities, similarly to the approach described in [52].

8.4.3 Distance Map Computation

To facilitate fast collision checking for the trajectory, we compute the Euclidean distance transform (EDT) of the occupancy volume. This way, a drone approximated by the bounding sphere can be checked for collision by one look-up query. We use an efficient $O(n)$ algorithm written by Felzenszwalb and Huttenlocher [38] to compute EDT of the volume, where $n = N^3$ is the number of voxels in the grid (the complexity is cubic in the size of the volume along a single axis). For querying distance and computing gradient, trilinear interpolation is used.

8.5 Trajectory Optimization

The local replanning problem is represented as an optimization of the following cost function:

$$E_{total} = E_{ep} + E_c + E_q + E_l, \tag{8.14}$$

where E_{ep} is an endpoint cost function that penalizes position and velocity deviations at the end of the optimized trajectory segment from the desired values that usually come from the global trajectory; E_c is a collision cost function; E_q is the cost of the integral over the squared derivatives (acceleration, jerk, snap); and E_l is a soft limit on the norm of time derivatives (velocity, acceleration, jerk and snap) over the trajectory.

8.5.1 Endpoint Cost Function

The purpose of the endpoint cost function is to keep the local trajectory close to the global one. This is achieved by penalizing position and velocity deviation at the end of the optimized trajectory segment from the desired values that come from the global trajectory. Because the property is formulated as a soft constraint, the targeted values might not be achieved, for example, because of obstacles blocking the path. The function is defined as follows:

$$E_{ep} = \lambda_p(p(t_{ep}) - p_{ep})^2 + \lambda_v(p'(t_{ep}) - p'_{ep})^2, \tag{8.15}$$

where t_{ep} is the end time of the segment, $p(t)$ is the trajectory to be optimized, p_{ep} and p'_{ep} are the desired position and velocity, respectively, λ_p and λ_v are the weighting parameters.

8.5.2 Collision Cost Function

Collision cost penalizes the trajectory points that are within the threshold distance τ to the obstacles. The cost function is computed as the following line integral:

$$E_c = \lambda_c \int_{t_{min}}^{t_{max}} c(p(t))||p'(t)||dt, \tag{8.16}$$

where the cost function for every point $c(x)$ is defined as follows:

$$c(x) = \begin{cases} \frac{1}{2\tau}(d(x) - \tau)^2 & \text{if } d(x) \leq \tau \\ 0 & \text{if } d(x) > \tau, \end{cases} \tag{8.17}$$

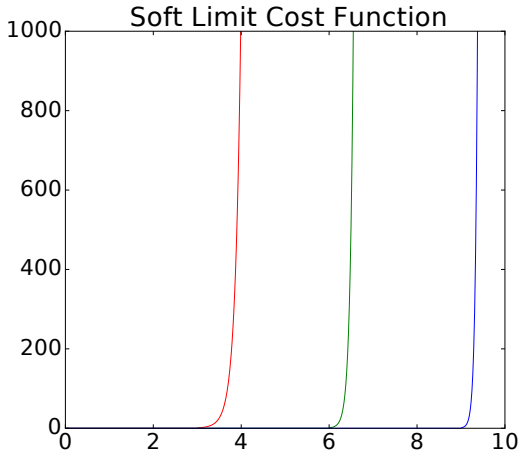


Figure 8.5: Soft limit cost function $l(x)$ proposed in Section 8.5.4 for p_{max} equals three (red), six (green), and nine (blue). This function acts as a soft limit on the time derivatives of the trajectory (velocity, acceleration, jerk, and snap) to ensure they are bounded and are feasible to execute by the MAV.

where τ is the distance threshold, $d(x)$ is the distance to the nearest obstacle, and λ_c is a weighting parameter. The line integral is computed using the rectangle method, and distances to the obstacles are obtained from the precomputed EDT by using trilinear interpolation.

8.5.3 Quadratic Derivative Cost Function

Quadratic derivative cost is used to penalize the integral over square derivatives of the trajectory (acceleration, jerk, and snap), and is defined as follows:

$$E_{ep} = \sum_{i=2}^{4} \int_{t_{min}}^{t_{max}} \lambda_{qi} (p^{(i)}(t))^2 dt. \tag{8.18}$$

The above function has a closed-form solution for trajectory segments represented using B-splines.

8.5.4 Derivative Limit Cost Function

To make sure that the computed trajectory is feasible, we must ensure that velocity, acceleration and higher derivatives of position remain bounded. This requirement can be included into the optimization as a constraint $\forall t : p^{(k)}(t) < p^k_{max}$, but in the proposed approach, we formulate it as a soft constraint by using the following

function:

$$E_{ep} = \sum_{i=2}^{4} \int_{t_{min}}^{t_{max}} l(p^{(i)}(t))dt, \tag{8.19}$$

where $l(x)$ is defined as follows:

$$l(x) = \begin{cases} \exp((p^{(k)}(x))^2 - (p^k_{max})^2) - 1 & \text{if } p^{(k)}(x) > p^k_{max} \\ 0 & \text{if } p^{(k)}(x) \leq p^k_{max} \end{cases} \tag{8.20}$$

This allows us to minimize this cost function by using any algorithm designed for unconstrained optimization.

8.5.5 Implementation Details

To run the local replanning algorithm on the drone, we first compute a global trajectory by using the approach described in [113]. This gives us a polynomial spline trajectory that avoids all mapped obstacles. Thereafter, we initialize our replanning algorithm with six fixed control points at the beginning of the global trajectory and C control points that need to be optimized.

In every iteration of the algorithm we set the endpoint constraints (Sec. 8.5.1) to be the position and velocity at t_{ep} of the global trajectory. The collision cost (Sec. 8.5.2) of the trajectory is evaluated using a circular buffer that contains measurements obtained using the RGB-D camera mounted on the drone. The weights of quadratic derivatives cost (Sec. 8.5.3) are set to the same values as those used for global trajectory generation, and the limits (Sec. 8.5.4) are set 20% higher to ensure that the global trajectory is followed with the appropriate velocity while laterally deviating from it.

After optimization, the first control point from the points that were optimized is fixed and sent to the MAV position controller. Another control point is added to the end of the spline, which increases t_{ep} and moves the endpoint further along the global trajectory.

For optimization we use [57], which provides an interface to several optimization algorithms. We have tested the MMA [125] and BFGS [75] algorithms for optimization, with both algorithms yielding similar performance.

8.6 Results

In this section, we present experimental results obtained using the proposed approach. First, we evaluate the mapping and the trajectory optimization components of the system separately for comparison with other approaches and justify their selection. Second, we evaluate the entire system in a realistic simulator in several

Algorithm	**Success Fraction**	**Mean Norm. Path Length**	**Mean Compute time** [s]
Inf. RRT* + Poly	**0.9778**	**1.1946**	2.2965
RRT Connect + Poly	0.9444	1.6043	**0.5444**
CHOMP N = 10	0.3222	1.0162	0.0032
CHOMP N = 100	0.5000	1.0312	0.0312
CHOMP N = 500	0.3333	1.0721	0.5153
[101] S = 2 jerk	0.4889	1.1079	**0.0310**
[101] S = 3 vel	0.4778	1.1067	0.0793
[101] S = 3 jerk	0.5000	1.0996	0.0367
[101] S = 3 jerk + Restart	**0.6333**	1.1398	0.1724
[101] S = 3 snap + Restart	0.6222	1.1230	0.1573
[101] S = 3 snap	0.5000	**1.0733**	0.0379
[101] S = 4 jerk	0.5000	1.0917	0.0400
[101] S = 5 jerk	0.5000	1.0774	0.0745
Ours C = 2	0.4777	**1.0668**	**0.0008**
Ours C = 3	0.4777	1.0860	0.0011
Ours C = 4	0.4888	1.1104	0.0015
Ours C = 5	0.5111	1.1502	0.0021
Ours C = 6	0.5555	1.1866	0.0028
Ours C = 7	0.5222	1.2368	0.0038
Ours C = 8	0.4777	1.2589	0.0054
Ours C = 9	**0.5777**	1.3008	0.0072

Table 8.1: Comparison of different path planning approaches. All results except thouse of the presented study are taken from [101]. Our approach performs similarly to approaches using polynomial splines without restarts, which indicates that B-splines can represent trajectories similar to those represented by polynomial splines. Lower computation times of our approach can be explained by the fact that unconstrained optimization occurs directly on the control points, unlike other approaches where the problem must first be transformed into an unconstrained form.

different environments, and finally, present an evaluation of the system running on real hardware.

8.6.1 Three-Dimensional Circular Buffer Performance

We compare our implementation of the 3D circular buffer to the popular octree-based solution of [52]. Both approaches use the same resolution of 0.1 m. We insert the depth maps sub-sampled to the resolution of 160×120, which come from a real-world dataset [124]. The results (Fig. 8.2) show that insertion of the data is more than an order of magnitude faster with the circular buffer, but only a limited space can be mapped with this approach. Because we need the map of a bounded neighborhood around the drone for local replanning, this drawback is not significant for target application.

8.6.2 Optimization Performance

To evaluate the trajectory optimization we use the forest dataset from [101]. Each spline configuration is tested in 9 environments with 10 random start and end positions that are at least 4 m away from each other. Each environment is $10 \times 10 \times 10$ m^3 in size and is populated with trees with increasing density. The optimization is initialized with a straight line and after optimization, we check for collisions. For all the approaches, the success fraction, mean normalized path length, and computation time are reported (Table 8.1).

The results of the proposed approach are similar in terms of success fraction to those achieved with polynomial splines from [101] without restarts, but the computation times with the proposed approach are significantly shorter. This is because the unconstrained optimization employed herein directly optimizes the control points, while in [101], a complicated procedure to transform a problem to the unconstrained optimization form [113] must be applied.

Another example of the proposed approach for trajectory optimization is shown in Figure 8.3, where a global trajectory is generated through a pre-defined set of points with an obstacle placed in the middle. The optimization is performed as described in Section 8.5.5, with the collision threshold τ set to 0.5 m. As can be seen in the plot, the local trajectory in the collision free regions aligns with the global one, but when an obstacle is encountered, a smooth trajectory is generated to avoid it and ensure that the MAV returns to the global trajectory.

8.6.3 System Simulation

To further evaluate our approach, we perform a realistic simulation experiment by using the Rotors simulator [44]. The main source of observations of the obstacles is a simulated RGB-D camera that produces VGA depth maps at 20 FPS. To control the

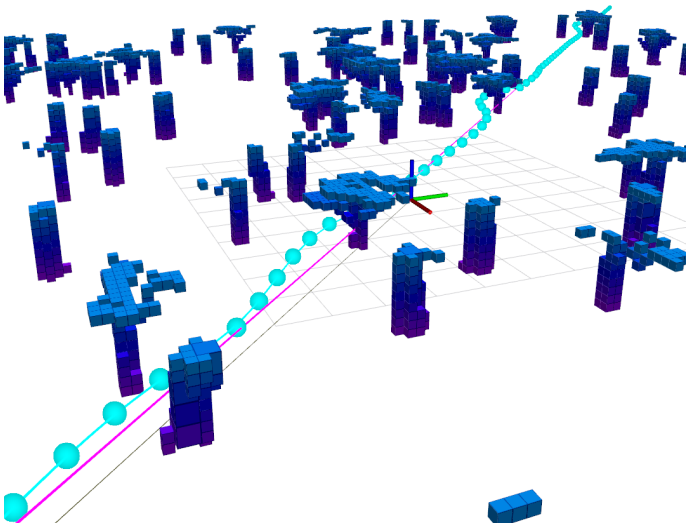


Figure 8.6: Result of local trajectory replanning algorithm running in a simulator on the forest dataset. The global trajectory is visualized in purple, local trajectory is represented as a uniform quintic B-spline, and its control points are visualized in cyan. Ground-truth octomap forest model is shown for visualization purposes.

MAV, we use the controller developed by Lee et al. [72], which is provided with the simulator and modified to receive trajectory messages as control points for uniform B-splines. When there are no new commands with control points, the last available control point is duplicated and inserted into the B-spline. This is useful from the viewpoint of failure case because when an MAV does not receive new control points, it will slowly stop at the last received control point.

We present the qualitative results of the simulations shown in Figures 8.1 and 8.6. The drone is initialized in free space and a global path through the world populated with obstacles is computed. In this case, the global path is computed to intersect the obstacles intensionally. The environment around the drone is mapped by inserting RGB-D measurements into the circular buffer, which is then used in the optimization procedure described above.

In all presented simulation experiments, the drone can compute a local trajectory that avoids collisions and keeps it close to the global path. The timings of the various operations involved in trajectory replanning are presented in Table 8.2.

8.6.4 Real-World Experiments

We evaluate our system on a multicopter in several outdoor experiments (Fig. 8.4). In these experiments, the drone is initialized without prior knowledge of the map and the global path is set as a straight line with its endpoint in front of the drone 1 m above the ground. The drone is required to use onboard sensors to map the

Operation	Computing 3D points	Moving volume	Inserting measurements	SDF computation	Trajectory optimization
Time [ms]	0.265	0.025	0.518	9.913	3.424

Table 8.2: Mean computation time for operations involved in trajectory replanning in the simulation experiment with depth map measurements sub-sampled to 160×120 and seven control points optimized.

environment and follow the global path avoiding trees, which serve as obstacles.

We use the AscTec Neo platform equipped with a stereo camera for estimating drone motion and an RGB-D camera (Intel Realsense R200) for obstacle mapping. All computations are performed on the drone on a 2.1 GHz Intel i7 CPU.

In all presented experiments, the drone can successfully avoid the obstacles and reach the goal position. However, the robustness of the system is limited at the moment owing to the accuracy of available RGB-D cameras that can capture outdoor scenes.

8.7 Conclusion

In this paper, we presented an approach to real-time local trajectory replanning for MAVs. We assumed that the global trajectory computed by an offline algorithm is provided and formulated an optimization problem that replans the local trajectory to follow the global one while avoiding unmodeled obstacles.

We improved the optimization performance by representing the local trajectory with uniform B-splines, which allowed us to perform unconstrained optimization and reduce the number of optimized parameters.

For collision checking we used the well-known concept of circular buffer to map a fixed area around the MAV, which improved the insertion times by an order of magnitude compared to those achieved with an octree-based solution.

In addition, we presented an evaluation of the complete system and specific sub-systems in realistic simulations and on real hardware.

Part III

Conclusion

Chapter 9

Summary

In this thesis we proposed several technologies relevant for autonomous vehicle navigation. We concentrated on the combination of cameras and IMUs as sensors for navigation, because of their low cost, small weight, size and power consumption. First, we proposed a novel camera model for cameras with fisheye lenses. Then, we presented two direct visual-inertial odometry methods and provided a visual-inertial dataset. After that, we proposed a novel method for real-time trajectory replanning and demonstrated autonomous vehicle navigation with a micro aerial vehicle as an example.

In the following, we make a short summary of the main contribution of each publication included in this thesis.

Chapter 4: We proposed the novel Double Sphere camera model, which works well with fisheye cameras, and compared it to other state-of-the-art camera models. We performed an extensive evaluation of the presented camera models on 16 different calibration sequences collected with 6 different lenses. The evaluations have shown that models based on higher-order polynomials, such as the Kannala-Brandt model, have the lowest reprojection error but are five to ten times slower than competing models. Both the proposed Double Sphere and Extended Unified models show very small reprojection error, with the Double Sphere model being slightly more accurate and the Extended Unified model being a bit faster. In addition, both models do not require computationally intensive trigonometric operations and have a closed-form inverse.

We showed that models based on spherical projection are a good alternative to models based on higher-order polynomials for applications requiring fast projection, unprojection, and a closed-form inverse.

Chapter 5: We introduced a novel approach to direct, tightly integrated visual-inertial odometry. It combines direct visual odometry that minimizes the reprojection error based on the estimated semi-dense depth maps with pre-integrated

IMU terms based on the measurements between consecutive frames. The system estimates depth maps and tracks the pose of new frames in an alternating fashion in two separate optimization processes. The two sensor sources complement each other ideally: stereo vision enables the system to correct the long-term IMU bias drift, while short-term IMU constraints help to cope with non-convexities in the photometric tracking formulation. This allows to track the system through large inter-frame motions or intervals without visual information. The method proposed in this chapter provides accurate semi-dense 3D reconstructions of the environment and outperforms existing feature-based approaches in terms of tracking accuracy.

Chapter 6: We presented a novel formulation of direct sparse visual-inertial odometry. Unlike the method presented in the previous chapter, we optimize both motion parameters and 3D geometry in a single optimization using camera and IMU measurements. This allows us to use a single monocular camera as the scale can be recovered from the IMU. The model explicitly includes scale and gravity direction in the model in order to deal with cases where the scale is not observable from the beginning. As the initial scale can be far from the true value we have proposed a novel technique called dynamic marginalization, where multiple marginalization priors are maintained to constrain the maximum scale difference. The quantitative evaluation demonstrated that the proposed visual-inertial odometry method outperforms the state-of-the-art and the approach presented in Chapter 5. In particular, experiments confirm that the inertial information provides a reliable scale estimate and significantly increases robustness and precision.

Chapter 7: We proposed a novel dataset for evaluating visual-inertial odometry with a diverse set of sequences in indoor and outdoor environments. It contains high resolution images with high dynamic range and vignette calibration, hardware time-synchronized with 3-axis accelerometer and gyro measurements. The dataset also contains accurate pose ground truth at high frequency at the start and end of each sequence. To have the ground truth data geometrically and temporally aligned with the IMU we perform hand-eye calibration on calibration sequences and time-offset estimation on all sequences. The dataset is publicly available with raw and calibrated data, as well as sequences to calibrate IMU white noise and random walk and vignetting of the camera.

We also evaluated the performance of state-of-the-art monocular and stereo visual-inertial methods on our benchmark. The results show that even for the best methods there is a significant drift on long indoor and outdoor sequences. This suggests that the dataset can be useful for the research community to further improve visual-inertial odometry methods.

Chapter 8: We introduced a real-time local trajectory replanning approach for micro aerial vehicles. Given the global trajectory computed by an offline algorithm we formulated an optimization problem that replans the local trajectory to avoid unmodeled obstacles and closely follow the global path.

By representing the local trajectory with uniform B-splines we improved the optimization performance by formulating the problem as an unconstrained optimization and reducing the number of optimized parameters.

The well-known concept of circular buffer was used for collision checking in order to map a fixed area around the MAV. This improved the measurement insertion time by an order of magnitude compared to the time of an octree-based solution.

Finally, we presented an evaluation of the complete system in realistic simulations and on a real micro aerial vehicle.

Chapter 10

Future Research

In this chapter we discuss several interesting research directions. We start from straightforward extensions that can be implemented to improve different aspects of the system and continue with long-term open challenges.

Using better camera models is a straightforward extension for the methods that were proposed in Chapters 5 and 6. Currently both implementations rely on pinhole rectified images that limit the field of view and introduce undistortion artefacts. One step in this direction was done in [2] (Figure 10.1), where the unified camera model was integrated into DSO, but we anticipate better results when using the models from Chapter 4.

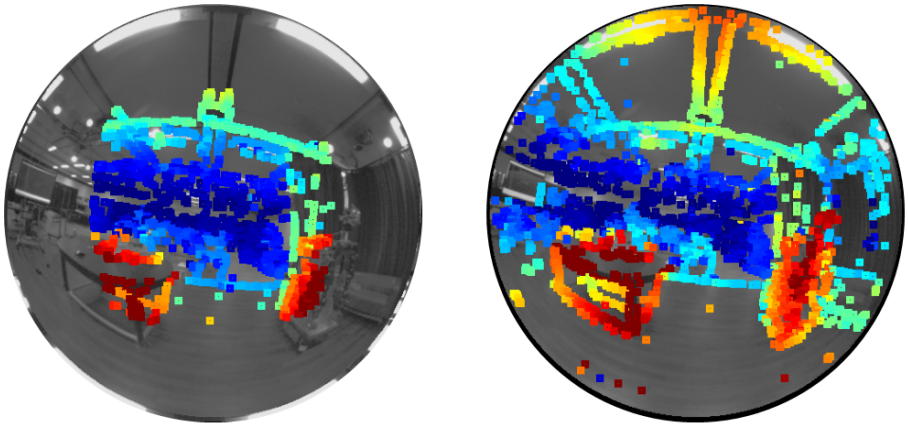


Figure 10.1: Comparison of pixels used for motion estimation for pinhole-rectified (left) and omnidirectional (right) camera models. Illustration taken from [2].

Better modeling of IMU scale, bias and axis misalignment. The motion estimation methods presented in Chapters 5 and 6 assume well calibrated IMU measurements, which are corrupted with white noise and slowly evolving bias. This

assumption holds well for industry-grade IMUs, for example the ADIS16448 in the Euroc dataset [20], but low-cost IMUs, that are typically used in smartphones, suffer from many other effects. Scaling of the axes, axis misalignment, rotation between accelerometer and gyroscope can be included into the optimization and estimated in real time. This can increase the computation time for the algorithms but may result in better accuracy.

Incorporating semantic information is another promising direction for future work. One obvious application is dynamic object filtering. State-of-the-art methods show high accuracy for instance segmentation and even generate masks for objects [50]. This information can be easily incorporated into SLAM and odometry methods to improve keypoint selection methods and the quality of the maps. Given the semantic meaning of the objects we can avoid selecting keypoints on objects that are known to be dynamic (humans, cars), which will result in better tracking and cleaner maps (Figure 10.2).

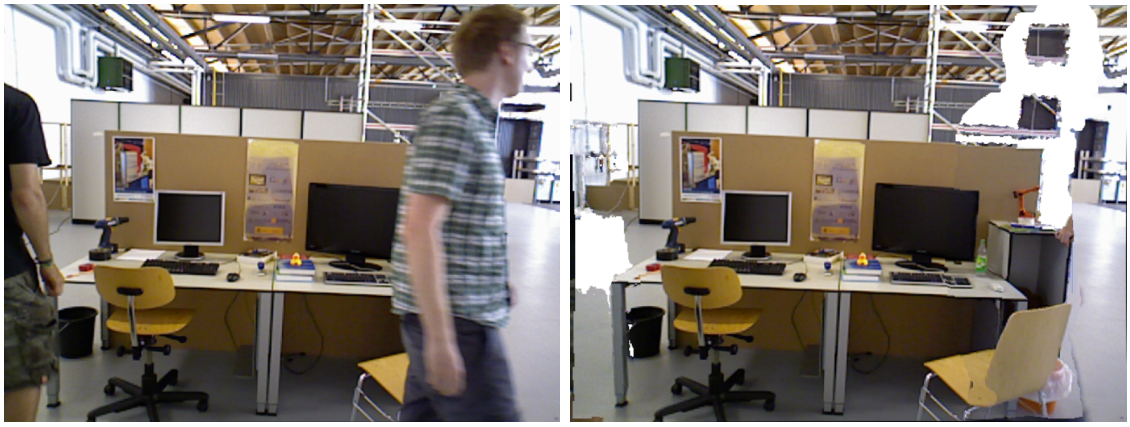


Figure 10.2: Semantic segmentation of the scene allows to remove dynamic objects to improve keypoint selection and mapping of the environment. Illustration taken from [15].

Long-term visual localization is a challenging task due to significant appearance variation between time of the day, season and weather. Currently even indirect methods for SLAM, that are more robust to illumination changes than direct methods, have significant problems to localize in the environment if the map was created in different light conditions. Learning based methods [76] [108] have shown impressive results for generating images for different seasons and times of the day that can enable localization for such challenging scenarios, but at the moment they rely on separate networks to generate images for every time of the day and weather condition (Figure 10.3).

Figure 10.3: Recent developments in deep learning for image-to-image translation. A winter-to-summer translation is shown on the top and a night-to-day translation is shown below. Illustrations taken from [76] and [108].

Finding an invariant representation for all possible variations in appearance or incorporating this knowledge directly at the keypoint level are two other directions that can improve vision-based localization.

Thorough Safety Analysis of Trajectory Generation Finally, for the obstacle avoidance and trajectory generation algorithms presented in Chapter 8 more thorough evaluation for practical applications is required. It is clear that the maximum allowed velocity of the vehicle should depend on the range of the sensor and the dimensions of the stored environment map. Otherwise, the path until full stop might be too long and result in a collision. Another aspect that requires further investigations is the field of view of the sensors. Zones that are not covered by the sensors can contain obstacles, so to guarantee an obstacle-free trajectory we have to ensure that the sensors cover well all directions where obstacles can appear.

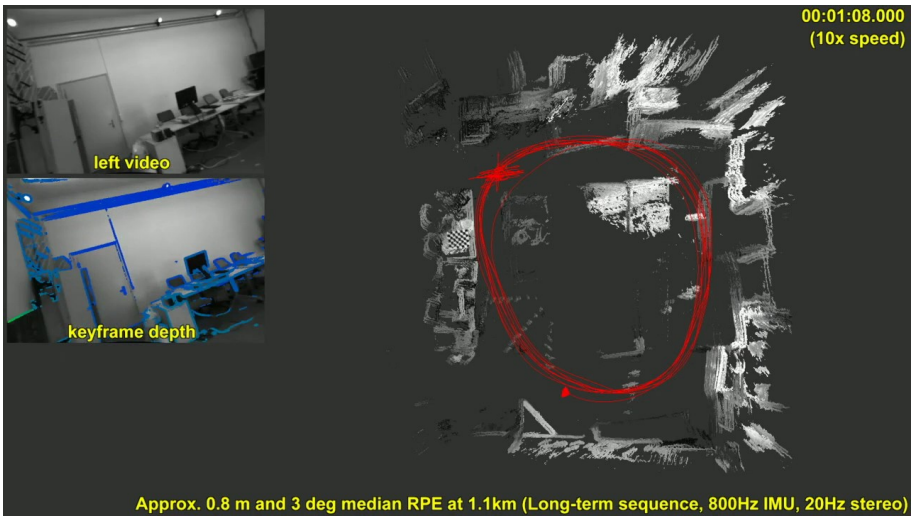
Part IV

Appendix

Appendix A

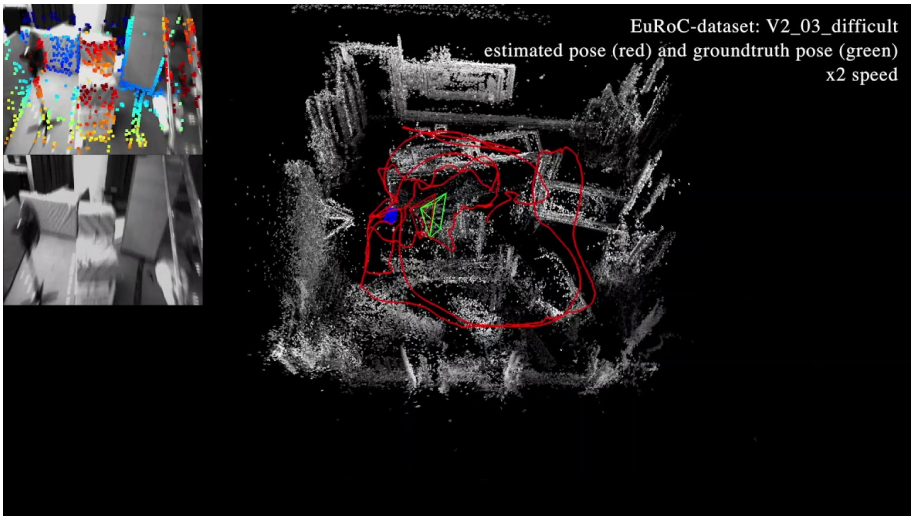
Multimedia Material

Direct Visual-Inertial Odometry with Stereo Cameras



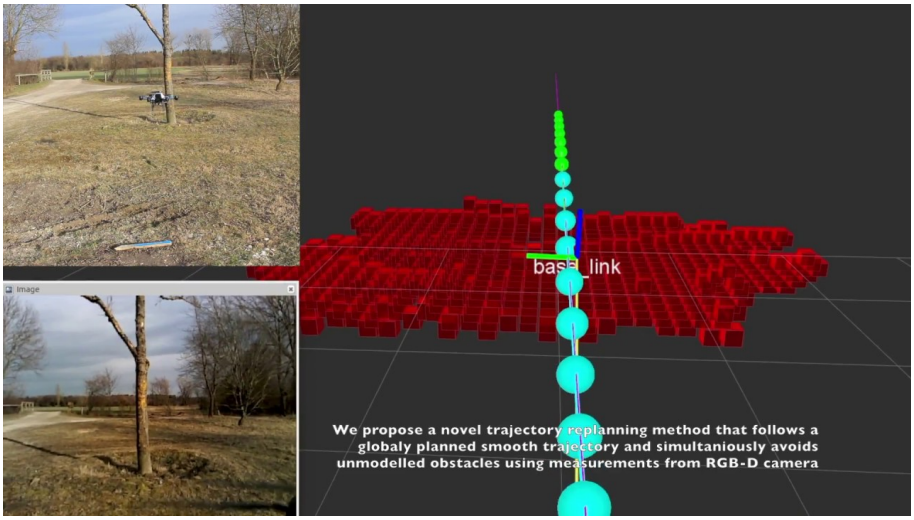
https://youtu.be/XSvFpPYfKWA

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization



https://youtu.be/GoqnXDS7jbA

Real-Time Trajectory Replanning for MAVs using Uniform B-splines and a 3D Circular Buffer



https://youtu.be/jh6tMHjxHSY

Appendix B

Open-Source Code and Datasets

The Double Sphere Camera Model

`https://vision.in.tum.de/research/vslam/double-sphere`: Dataset, calibration results and open-source implementation of the camera model.

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization

`https://vision.in.tum.de/research/vslam/vi-dso`: Additional evaluation results and supplementary material.

TUM VI Benchmark for Evaluating Visual-Inertial Odometry

`https://vision.in.tum.de/data/datasets/visual-inertial-dataset`: Dataset, calibration dataset, calibration results and evaluations.

Real-Time Trajectory Replanning for MAVs using Uniform splines and a 3D Circular Buffer

`https://github.com/VladyslavUsenko/ewok`: C++ implementation of the proposed algorithm with demonstration in realistic simulator.

List of Figures

1.1 Example of a 3D reconstruction 4
1.2 An example of direct visual-inertial odometry 6
1.3 Examples of 3D environment representations 8
1.4 Path planning approaches 9
1.5 Autonomous driving examples 10
1.6 An MAV inspection example 11

2.1 Setup for collecting the visual-inertial dataset 17
2.2 Autonomous navigation of a flying vehicle 18

4.1 Double Sphere camera model 40
4.2 Unified and Extended Unified camera models 41
4.3 Kannala-Brandt camera model 43
4.4 Field-of-View camera model 46
4.5 Lenses used to evaluate camera models 47
4.6 Corners of the calibration pattern projected onto the image 52

5.1 Tight IMU fusion results in more accurate position tracking 59
5.2 Factor graph representing the visual-inertial odometry 62
5.3 Evolution of the factor graph 64
5.4 Orientation error 67
5.5 Translational drift 69
5.6 Long-run comparison 71
5.7 Qualitative results on Malaga Urban dataset 72
5.8 Images from EuRoC and Malaga datasets 74

6.1 Results on the EuRoC-dataset 79
6.2 Factor graphs for the visual-inertial odometry 84
6.3 Partitioning of the factor graph 86
6.4 Scale estimation 87
6.5 RMSE on EuRoC dataset 89
6.6 Cumulative error plot 90
6.7 Scale estimate for MH_04_difficult 91
6.8 RPE on the EuRoC-dataset 95

7.1 TUM VI benchmark 99

7.2 Sensor setup . . . 103
7.3 Relation of Illuminance measurements to exposure . . . 104
7.4 Time alignment . . . 106
7.5 Allan deviation . . . 108
7.6 Qualitative results . . . 112

8.1 Example of local trajectory replanning . . . 119
8.2 Comparison between octomap and circular buffer . . . 122
8.3 Example of online trajectory replanning . . . 126
8.4 Real-world experiment . . . 127
8.5 Soft limit cost function . . . 130
8.6 Result of local trajectory replanning . . . 134

10.1 Pinhole-rectified and omnidirectional camera models . . . 143
10.2 Dynamic object filtering . . . 144
10.3 Image-to-image translation . . . 145

Own Publications

[1] J. Engel, V. Usenko and D. Cremers. "A Photometrically Calibrated Benchmark for Monocular Visual Odometry". In: *arXiv preprint arXiv:1607.02555* (2016). eprint: http://arxiv.org/abs/1607.02555v2 (cit. on pp. 14, 101, 107).

[2] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler and D. Cremers. "Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras". In: *IEEE Robotics and Automation Letters* (2018). DOI: 10.1109/lra.2018.2855443. eprint: http://arxiv.org/abs/1808.02775 (cit. on pp. 14, 143).

[3] D. Schubert, N. Demmel, V. Usenko, J. Stückler and D. Cremers. "Direct Sparse Odometry With Rolling Shutter". In: *Proc. of the European Conference on Computer Vision (ECCV)*. Sept. 2018. eprint: http://arxiv.org/abs/1808.00558 (cit. on p. 14).

[4] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler and D. Cremers. "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robot Systems (IROS)*. Oct. 2018. eprint: http://arxiv.org/abs/1804.06120v1 (cit. on pp. 13, 14, 97).

[5] L. von Stumberg, V. Usenko and D. Cremers. "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. May 2018. DOI: 10.1109/ICRA.2018.8462905. eprint: http://arxiv.org/abs/1804.05625v1 (cit. on pp. 13, 14, 77).

[6] L. von Stumberg, V. Usenko, J. Engel, J. Stückler and D. Cremers. "From Monocular SLAM to Autonomous Drone Exploration". In: *2017 European Conference on Mobile Robots (ECMR)*. IEEE, Sept. 2017. DOI: 10.1109/ecmr.2017.8098709. eprint: http://arxiv.org/abs/1609.07835v3 (cit. on pp. 14, 15).

[7] V. Usenko, N. Demmel and D. Cremers. "The Double Sphere Camera Model". In: *Proc. of the Int. Conference on 3D Vision (3DV)*. Sept. 2018. DOI: 10.1109/3DV.2018.00069. eprint: http://arxiv.org/abs/1807.08957 (cit. on pp. 13, 14, 37).

[8] V. Usenko, J. Engel, J. Stückler and D. Cremers. "Direct Visual-Inertial Odometry with Stereo Cameras". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. DOI: 10.1109/icra.2016.7487335 (cit. on pp. 7, 13, 14, 57, 80–82, 90, 91, 95, 98).

[9] V. Usenko, J. Engel, J. Stückler and D. Cremers. "Reconstructing Street-Scenes in Real-Time from a Driving Car". In: *2015 International Conference on 3D Vision*. IEEE, Oct. 2015. DOI: 10.1109/3dv.2015.75 (cit. on p. 14).

[10] 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: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8202160 (cit. on pp. 13, 14, 117).

Bibliography

[11] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli and R. Siegwart. "Motion- and Uncertainty-aware Path Planning for Micro Aerial Vehicles". In: *Journal of Field Robotics* 31.4 (June 2014), pp. 676–698. DOI: 10.1002/rob.21522 (cit. on pp. 118, 121).

[12] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz and R. Szeliski. "Building Rome in a Day". In: *Commun. ACM* 54.10 (Oct. 2011), pp. 105–112. ISSN: 0001-0782. DOI: 10.1145/2001269.2001293 (cit. on p. 4).

[13] J. Amanatides and A. Woo. "A Fast Voxel Traversal Algorithm for Ray Tracing". In: *Eurographics '87*. 1987 (cit. on p. 128).

[14] H. Bay, T. Tuytelaars and L. Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof and A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8. DOI: 10.1007/11744023_32 (cit. on p. 5).

[15] B. Bescos, J. M. Fácil, J. Civera and J. Neira. "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 4076–4083. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2860039 (cit. on p. 144).

[16] J.-L. Blanco-Claraco, F.-A. Moreno-Duenas and J. Gonzalez-Jimenez. "The Malaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario". In: *The International Journal of Robotics Research* 33.2 (2014), pp. 207–214. DOI: 10.1177/0278364913507326 (cit. on pp. 73, 100, 101).

[17] M. Bloesch, M. Burri, S. Omari, M. Hutter and R. Siegwart. "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback". In: *The International Journal of Robotics Research* 36.10 (Sept. 2017), pp. 1053–1072. DOI: 10.1177/0278364917728574 (cit. on pp. 7, 60, 80, 89, 90, 98, 113).

[18] C. de Boor. "On calculating with B-splines". In: *Journal of Approximation Theory* 6.1 (July 1972), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9 (cit. on p. 121).

[19] O. Bottema and B. Roth. *Theoretical Kinematics*. Dover Books on Physics. Dover Publications, 2012. ISBN: 9780486663463 (cit. on p. 24).

[20] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik and R. Siegwart. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (Jan. 2016), pp. 1157–1163. DOI: 10.1177/0278364915620033 (cit. on pp. 51, 79, 90, 98, 100, 101, 144).

[21] M. Burri, H. Oleynikova, M. W. Achtelik and R. Siegwart. "Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015. DOI: 10.1109/iros.2015.7353622 (cit. on pp. 9, 120).

[22] M. Calonder, V. Lepetit, C. Strecha and P. Fua. "BRIEF: Binary Robust Independent Elementary Features". In: *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1. DOI: 10.1007/978-3-642-15561-1_56 (cit. on p. 5).

[23] N. Carlevaris-Bianco, A. K. Ushani and R. M. Eustice. "University of Michigan North Campus long-term vision and lidar dataset". In: *International Journal of Robotics Research (IJRR)* 35.9 (2015), pp. 1023–1035. DOI: 10.1177/0278364915614638 (cit. on p. 100).

[24] 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: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2014. DOI: 10.1109/ICRA.2014.6907483 (cit. on p. 82).

[25] Y. Chen, F. Wu, W. Shuai and X. Chen. "Robots serve humans in public places—KeJia robot as a shopping assistant". In: *International Journal of Advanced Robotic Systems* 14.3 (2017), p. 1729881417703569. DOI: 10.1177/1729881417703569. eprint: https://doi.org/10.1177/1729881417703569 (cit. on p. 9).

[26] A. Chiuso, P. Favaro, H. Jin and S. Soatto. "Structure from motion causally integrated over time". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.4 (Apr. 2002), pp. 523–535. ISSN: 0162-8828. DOI: 10.1109/34.993559 (cit. on p. 5).

[27] A. Comport, E. Malis and P. Rives. "Accurate Quadri-focal Tracking for Robust 3D Visual Odometry". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2007. DOI: 10.1109/robot.2007.363762 (cit. on pp. 6, 60, 80).

[28] M. G. Cox. "The numerical evaluation of B-splines". In: *IMA Journal of Applied Mathematics* (1972). DOI: 10.1093/imamat/10.2.134 (cit. on p. 121).

[29] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (June 2007), pp. 1052–1067. DOI: 10.1109/tpami.2007.1049 (cit. on pp. 5, 80).

[30] F. Devernay and O. Faugeras. "Straight lines have to be straight". In: *Machine vision and applications* 13.1 (2001), pp. 14–24. DOI: 10.1007/pl00013269 (cit. on pp. 46, 48).

[31] D. Dolgov, S. Thrun, M. Montemerlo and J. Diebel. "Practical search techniques in path planning for autonomous driving". In: *Ann Arbor* (2008) (cit. on pp. 9, 120).

[32] J. Engel, V. Koltun and D. Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (Mar. 2018), pp. 611–625. DOI: 10.1109/tpami.2017.2658577 (cit. on pp. 6, 16, 19, 78, 80, 81, 83, 86, 89, 90, 98).

[33] J. Engel, T. Schöps and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 834–849. DOI: 10.1007/978-3-319-10605-2_54 (cit. on pp. 6, 15, 19, 60, 62–65, 72, 80).

[34] J. Engel, J. Stückler and D. Cremers. "Large-scale direct SLAM with stereo cameras". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015. DOI: 10.1109/iros.2015.7353631 (cit. on pp. 59, 62–64).

[35] J. Engel, J. Sturm and D. Cremers. "Camera-based navigation of a low-cost quadrocopter". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012. DOI: 10.1109/iros.2012.6385458 (cit. on pp. 7, 60, 80).

[36] J. Engel, J. Sturm and D. Cremers. "Semi-dense Visual Odometry for a Monocular Camera". In: *2013 IEEE International Conference on Computer Vision*. IEEE, Dec. 2013. DOI: 10.1109/iccv.2013.183 (cit. on p. 70).

[37] R. M. Eustice, H. Singh and J. J. Leonard. "Exactly sparse delayed-state filters for view-based SLAM". In: *IEEE Transactions on Robotics (TRO)* 22.6 (2006), pp. 1100–1114. DOI: 10.1109/tro.2006.886264 (cit. on p. 60).

[38] P. F. Felzenszwalb and D. P. Huttenlocher. "Distance Transforms of Sampled Functions". In: *Theory of Computing* (2012) (cit. on p. 128).

[39] 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 XI*. Robotics: Science and Systems Foundation, July 2015. DOI: 10.15607/rss.2015.xi.006 (cit. on pp. 7, 60, 80–82).

[40] C. Forster, M. Pizzoli and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014. DOI: 10.1109/icra.2014.6906584 (cit. on pp. 6, 60, 80).

[41] D. Fox, W. Burgard and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics Automation Magazine* 4.1 (Mar. 1997), pp. 23–33. ISSN: 1070-9932. DOI: 10.1109/100.580977 (cit. on p. 9).

[42] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik and M. Pollefeys. "Building Rome on a Cloudless Day". In: *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 368–381. ISBN: 978-3-642-15561-1. DOI: 10.1007/978-3-642-15561-1_27 (cit. on p. 4).

[43] P. Furgale, J. Rehder and R. Siegwart. "Unified temporal and spatial calibration for multi-sensor systems". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. IEEE, Nov. 2013, pp. 1280–1286. DOI: 10.1109/iros.2013.6696514 (cit. on pp. 45, 72, 111).

[44] F. Furrer, M. Burri, M. Achtelik and R. Siegwart. "Robot Operating System (ROS)". In: *Studies in Computational Intelligence* (2016) (cit. on p. 133).

[45] A. Geiger, P. Lenz and R. Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2012. DOI: 10.1109/cvpr.2012.6248074 (cit. on pp. 58, 78).

[46] A. Geiger, P. Lenz, C. Stiller and R. Urtasun. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (Aug. 2013), pp. 1231–1237. DOI: 10.1177/0278364913491297 (cit. on pp. 100, 101).

[47] C. Geyer and K. Daniilidis. "A Unifying Theory for Central Panoramic Systems and Practical Implications". In: *Computer Vision — ECCV 2000*. Ed. by D. Vernon. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 445–461. ISBN: 978-3-540-45053-5. DOI: 10.1007/3-540-45053-x_29 (cit. on p. 42).

[48] J. Gui, D. Gu and H. Hu. "Robust direct visual inertial odometry via entropy-based relative pose estimation". In: *Proc. of the IEEE Int. Conf. on Mechatronics and Automation (ICMA)*. 2015. DOI: 10.1109/icma.2015.7237603 (cit. on p. 60).

[49] C. Harris and M. Stephens. "A combined corner and edge detector". In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151. DOI: 10.5244/C.2.23 (cit. on p. 5).

[50] K. He, G. Gkioxari, P. Dollár and R. Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322 (cit. on p. 144).

[51] L. Heng, G. H. Lee and M. Pollefeys. "Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle". In: *Autonomous robots* 39.3 (2015), pp. 259–277. DOI: 10.1007/s10514-015-9466-8 (cit. on p. 50).

[52] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss and W. Burgard. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Autonomous Robots* (2013). DOI: 10.1007/s10514-012-9321-0 (cit. on pp. 8, 121, 128, 133).

[53] *IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros*. 1998. DOI: 10.1109/ieeestd.1998.86153 (cit. on p. 107).

[54] V. Indelman, S. Williams, M. Kaess and F. Dellaert. "Information Fusion in Navigation Systems via Factor Graph Based Incremental Smoothing". In: *Int. Journal of Robotics and Autonomous Systems (RAS)* 61.8 (2013), pp. 721–738. DOI: 10.1016/j.robot.2013.05.001 (cit. on p. 66).

[55] M. Irani and P. Anandan. "Direct Methods". In: *Vision Algorithms: Theory and Practice, LNCS 1883*. Springer, 2002. DOI: 10.1017/cbo9780511599866.004 (cit. on p. 60).

[56] H. Jin, P. Favaro and S. Soatto. "Real-time 3D Motion and Structure of Point Features: a Front-end System for Vision-based Control and Interaction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000. DOI: 10.1109/cvpr.2000.854954 (cit. on p. 98).

[57] S. G. Johnson. "The NLopt nonlinear-optimization package". In: () (cit. on p. 131).

[58] E. S. Jones and S. Soatto. "Visual-inertial Navigation, Mapping and Localization: A Scalable Real-time Causal Approach". In: *The International Journal of Robotics Research* 30.4 (Jan. 2011), pp. 407–430. DOI: 10.1177/0278364910388963 (cit. on p. 60).

[59] D. Jung and P. Tsiotras. "On-line path generation for small unmanned aerial vehicles using B-spline path templates". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2008. DOI: 10.2514/6.2008-7135 (cit. on pp. 9, 120).

[60] J. Kaiser, A. Martinelli, F. Fontana and D. Scaramuzza. "Simultaneous State Initialization and Gyroscope Bias Calibration in Visual Inertial Aided Navigation". In: *IEEE Robot. and Autom. Lett.* 2.1 (2017). ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2521413 (cit. on pp. 7, 81, 93).

[61] J. Kannala and S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.8 (Aug. 2006), pp. 1335–1340. DOI: 10.1109/tpami.2006.153 (cit. on pp. 39, 44, 48).

[62] K. Kashin. "Statistical Inference: Maximum Likelihood Estimation". In: (2014). eprint: http://www.konstantinkashin.com/notes/stat/Maximum_Likelihood_Estimation.pdf (cit. on p. 28).

[63] A. Kasyanov, F. Engelmann, J. Stückler and B. Leibe. "Keyframe-based visual-inertial online SLAM with relocalization". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8206581 (cit. on pp. 90–92).

[64] N. Keivan, A. Patron-Perez and G. Sibley. "Asynchronous Adaptive Conditioning for Visual-Inertial SLAM". In: *Int. Symposium on Experimental Robotics (ISER)*. 2014. DOI: 10.1007/978-3-319-23778-7_21 (cit. on p. 60).

[65] C. Kerl, J. Sturm and D. Cremers. "Dense visual SLAM for RGB-D cameras". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013. DOI: 10.1109/iros.2013.6696650 (cit. on p. 6).

[66] C. Kerl, J. Sturm and D. Cremers. "Robust odometry estimation for RGB-D cameras". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013. DOI: 10.1109/icra.2013.6631104 (cit. on pp. 6, 60, 78, 80).

[67] B. Khomutenko, G. Garcia and P. Martinet. "An Enhanced Unified Camera Model". In: *IEEE Robotics and Automation Letters* 1.1 (Jan. 2016), pp. 137–144. ISSN: 2377-3766. DOI: 10.1109/lra.2015.2502921 (cit. on pp. 43, 48).

[68] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, Nov. 2007. DOI: 10.1109/ismar.2007.4538852 (cit. on pp. 5, 60, 70, 80).

[69] L. Kneip, H. Li and Y. Seo. "Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability". In: *European Conference on Computer Vision*. Springer. 2014, pp. 127–142. DOI: 10.1007/978-3-319-10590-1_9 (cit. on p. 50).

[70] E. Kruppa. "Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung". In: *Sitzungsberichte der Mathematisch-Naturwissenschaftlichen Kaiserlichen Akademie der Wissenschaften* 122 (1913), pp. 1939–1948. eprint: http://arxiv.org/abs/1801.01454 (cit. on p. 4).

[71] Y. Kuznietsov, J. Stückler and B. Leibe. "Semi-Supervised Deep Learning for Monocular Depth Map Prediction". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 2215–2223. DOI: 10.1109/CVPR.2017.238 (cit. on p. 10).

[72] T. Lee, M. Leoky and N. H. McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3)". In: *Conference on Decision and Control*. 2010. DOI: 10.1109/cdc.2010.5717652 (cit. on p. 134).

[73] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart and P. Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (Dec. 2014), pp. 314–334. DOI: 10.1177/0278364914554813 (cit. on pp. 7, 60, 71, 73, 80, 81, 90–92, 98, 113).

[74] M. Li and A. I. Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (May 2013), pp. 690–711. DOI: 10.1177/0278364913481251 (cit. on pp. 7, 60, 80).

[75] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.1-3 (Aug. 1989), pp. 503–528. DOI: 10.1007/bf01589116 (cit. on p. 131).

[76] M. Liu, T. Breuel and J. Kautz. "Unsupervised Image-to-Image Translation Networks". In: *CoRR* abs/1703.00848 (2017). arXiv: 1703.00848 (cit. on pp. 144, 145).

[77] H. C. Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections". In: *Nature* 293.5828 (1981), p. 133. DOI: 10.1038/293133a0 (cit. on p. 4).

[78] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. Sept. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410 (cit. on p. 5).

[79] T. Lupton and S. Sukkarieh. "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions". In: *IEEE Transactions on Robotics* 28.1 (Feb. 2012), pp. 61–76. DOI: 10.1109/tro.2011.2170332 (cit. on pp. 66, 82).

[80] W. Maddern, G. Pascoe, C. Linegar and P. Newman. "1 year, 1000 km: The Oxford RobotCar dataset". In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498 (cit. on p. 101).

[81] K. Madson, H. B. Nielsen and O. Tingleff. "Methods for non-linear least squares problem". In: (cit. on pp. 30–32).

[82] M. Maimone, Y. Cheng and L. Matthies. "Two years of visual odometry on the mars exploration rovers". In: *Journal of Field Robotics* 24.3 (2007), pp. 169–186. DOI: 10.1002/rob.20184 (cit. on p. 5).

[83] E. Mair, G. D. Hager, D. Burschka, M. Suppa and G. Hirzinger. "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test". In: *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–196. ISBN: 978-3-642-15552-9. DOI: 10.1007/978-3-642-15552-9_14 (cit. on p. 5).

[84] A. L. Majdik, C. Till and D. Scaramuzza. "The Zurich urban micro aerial vehicle dataset". In: *The International Journal of Robotics Research (IJRR)* 36.3 (2017), pp. 269–273. DOI: 10.1177/0278364917702237 (cit. on pp. 100, 101).

[85] A. Martinelli. "Closed-Form Solution of Visual-Inertial Structure from Motion". In: *Int. Journal of Computer Vision (IJCV)* 106.2 (2014). ISSN: 1573-1405. DOI: 10.1007/s11263-013-0647-7 (cit. on pp. 7, 81, 83, 93).

[86] C. Mei and P. Rives. "Single View Point Omnidirectional Camera Calibration from Planar Grids". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, Apr. 2007, pp. 3945–3950. DOI: 10.1109/robot.2007.364084 (cit. on pp. 42, 48, 53, 55).

[87] L. Meier, P. Tanskanen, F. Fraundorfer and M. Pollefeys. "PIXHAWK: A system for autonomous flight using onboard computer vision". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. DOI: 10.1109/icra.2011.5980229 (cit. on pp. 7, 58, 60, 80).

[88] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. DOI: 10.1109/icra.2011.5980409 (cit. on pp. 118, 121).

[89] O. Miksik, P. H. Torr, V. Vineet, M. Lidegaard, R. Prasaath, M. Nießner, S. Golodetz, S. L. Hicks, P. Pérez and S. Izadi. "The Semantic Paintbrush". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, 2015. DOI: 10.1145/2702123.2702222 (cit. on p. 58).

[90] H. P. Moravec. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover". AAI8024717. PhD thesis. Stanford, CA, USA, 1980 (cit. on pp. 5, 10).

[91] P. Moulon, P. Monasse, R. Marlet et al. *OpenMVG. An Open Multiple View Geometry library.* https://github.com/openMVG/openMVG (cit. on p. 4).

[92] A. I. Mourikis and S. I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, Apr. 2007, pp. 3565–3572. DOI: 10.1109/robot.2007.364024 (cit. on p. 73).

[93] M. W. Mueller, M. Hehn and R. D'Andrea. "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013. DOI: 10.1109/iros.2013.6696852 (cit. on pp. 9, 120).

[94] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163. ISSN: 1552-3098. DOI: 10.1109/tro.2015.2463671. eprint: http://arxiv.org/abs/1502.00956v2 (cit. on pp. 5, 80, 90, 91, 93, 95).

[95] R. Mur-Artal and J. D. Tardos. "Visual-Inertial Monocular SLAM With Map Reuse". In: *IEEE Robotics and Automation Letters* 2.2 (Apr. 2017), pp. 796–803. DOI: 10.1109/lra.2017.2653359. eprint: http://arxiv.org/abs/1610.05949v2 (cit. on pp. 7, 80, 81, 83, 84).

[96] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim and A. Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, Oct. 2011. DOI: 10.1109/ismar.2011.6092378 (cit. on pp. 6, 60).

[97] R. A. Newcombe, S. J. Lovegrove and A. J. Davison. "DTAM: Dense tracking and mapping in real-time". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126513 (cit. on pp. 6, 60, 80).

[98] M. Nießner, M. Zollhöfer, S. Izadi and M. Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Transactions on Graphics* 32.6 (Nov. 2013), pp. 1–11. DOI: 10.1145/2508363.2508374 (cit. on pp. 8, 121).

[99] D. Nister. "An efficient solution to the five-point relative pose problem". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. June 2003, pp. II–195. DOI: 10.1109/CVPR.2003.1211470 (cit. on p. 5).

[100] D. Nister, O. Naroditsky and J. Bergen. "Visual odometry". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. IEEE, 2004. DOI: 10.1109/cvpr.2004.1315094 (cit. on pp. 5, 80).

[101] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart and E. Galceran. "Continuous-time trajectory optimization for online UAV replanning". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016. DOI: 10.1109/iros.2016.7759784 (cit. on pp. 9, 119, 120, 124, 132, 133).

[102] H. Oleynikova, Z. Taylor, M. Fehr, J. Nieto and R. Siegwart. "Voxblox: Building 3D Signed Distance Fields for Planning". In: *arXiv preprint arXiv:1611.03631* (2016) (cit. on pp. 8, 121).

[103] C. F. Olson, L. H. Matthies, H. Schoppers and M. W. Maimone. "Robust stereo ego-motion for long distance navigation". In: *IEEE Int. Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000. DOI: 10.1109/cvpr.2000.854879 (cit. on p. 98).

[104] E. Olson. "AprilTag: A robust and flexible visual fiducial system". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. DOI: 10.1109/icra.2011.5979561 (cit. on pp. 49, 102).

[105] A. Paranjape, K. C. Meier, X. Shi, S.-J. Chung and S. Hutchinson. "Motion primitives and 3D path planning for fast flight through a forest". In: *The International Journal of Robotics Research* (2015). DOI: 10.1109/iros.2013.6696773 (cit. on pp. 9, 120).

[106] Y. Pawitan. *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford science publications. OUP Oxford, 2001. ISBN: 9780198507659 (cit. on p. 27).

[107] B. Pfrommer, N. Sanket, K. Daniilidis and J. Cleveland. "PennCOSYVIO: A challenging Visual Inertial Odometry benchmark". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. DOI: 10.1109/icra.2017.7989443 (cit. on pp. 100, 101).

[108] H. Porav, W. Maddern and P. Newman. "Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer". In: *CoRR* abs/1803.03341 (2018). arXiv: 1803.03341 (cit. on pp. 144, 145).

[109] K. Qin. "General matrix representations for B-splines". In: *The Visual Computer* 16.3-4 (May 2000), pp. 177–186. DOI: 10.1007/s003710050206 (cit. on p. 121).

[110] T. Qin, P. Li and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *arXiv preprint arXiv:1708.03852* (2017). DOI: 10.1109/tro.2018.2853729 (cit. on p. 113).

[111] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690 (cit. on p. 10).

[112] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann and R. Siegwart. "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. DOI: 10.1109/icra.2016.7487628 (cit. on p. 106).

[113] C. Richter, A. Bry and N. Roy. "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments". In: *Springer Tracts in Advanced Robotics*. Springer International Publishing, 2016, pp. 649–666. DOI: 10.1007/978-3-319-28872-7_37 (cit. on pp. 9, 118, 120, 124, 131, 133).

[114] A. Rituerto, L. Puig and J. Guerrero. "Comparison of omnidirectional and conventional monocular systems for visual slam". In: *10th OMNIVIS with RSS*. 2010 (cit. on p. 38).

[115] E. Rosten, R. Porter and T. Drummond. "Faster and Better: A Machine Learning Approach to Corner Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (Jan. 2010), pp. 105–119. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.275 (cit. on p. 5).

[116] D. Scaramuzza, A. Martinelli and R. Siegwart. "A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion". In: *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*. IEEE, 2006, pp. 45–45. DOI: 10.1109/icvs.2006.3 (cit. on pp. 39, 45).

[117] T. B. Schön and F. Lindsten. "Manipulating the multivariate gaussian density". In: (2011). eprint: http://users.isy.liu.se/en/rt/schon/Publications/SchonL2011.pdf (cit. on p. 29).

[118] J. L. Schönberger and J. Frahm. "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 4104–4113. DOI: 10.1109/CVPR.2016.445 (cit. on p. 4).

[119] J. Shi and Tomasi. "Good features to track". In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. June 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794 (cit. on p. 5).

[120] N. Snavely, S. M. Seitz and R. Szeliski. "Modeling the World from Internet Photo Collections". In: *International Journal of Computer Vision* 80.2 (Nov. 1, 2008), pp. 189–210. ISSN: 1573-1405. DOI: 10.1007/s11263-007-0107-3 (cit. on p. 4).

[121] H. Sommer, I. Gilitschenski, M. Bloesch, S. Weiss, R. Siegwart and J. I. Nieto. "Why and How to Avoid the Flipped Quaternion Multiplication". In: *CoRR* abs/1801.07478 (2018). arXiv: 1801.07478 (cit. on p. 22).

[122] F. Steinbrucker, J. Sturm and D. Cremers. "Volumetric 3D mapping in real-time on a CPU". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014. DOI: 10.1109/icra.2014.6907127 (cit. on pp. 8, 121).

[123] A. Stelzer, H. Hirschmüller and M. Görner. "Stereo-vision-based Navigation of a Six-legged Walking Robot in Unknown Rough Terrain". In: *Int. Journal of Robotics Research (IJRR)* (2012). DOI: 10.1177/0278364911435161 (cit. on pp. 58, 78).

[124] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. IEEE, Oct. 2012, pp. 573–580. DOI: 10.1109/iros.2012.6385773 (cit. on pp. 101, 111, 122, 133).

[125] K. Svanberg. "A Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations". In: *SIAM Journal on Optimization* 12.2 (Jan. 2002), pp. 555–573. DOI: 10.1137/s1052623499362822 (cit. on p. 131).

[126] P. Tanskanen, T. Naegeli, M. Pollefeys and O. Hilliges. "Semi-direct EKF-based monocular visual-inertial odometry". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015, pp. 6073–6078. DOI: 10.1109/iros.2015.7354242 (cit. on pp. 7, 60).

[127] M. R. Walter, R. M. Eustice and J. J. Leonard. "Exactly Sparse Extended Information Filters for Feature-based SLAM". In: *The International Journal of Robotics Research* 26.4 (2007), pp. 335–359. DOI: 10.1177/0278364906075026. eprint: https://doi.org/10.1177/0278364906075026 (cit. on p. 29).

[128] R. Wang, M. Schworer and D. Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.421 (cit. on p. 98).

[129] Waymo. *Waymo Safety Report. On the Road to Fully Self-Driving.* https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017.pdf. 2017 (cit. on p. 10).

[130] S. Weiss, M. Achtelik, S. Lynen, M. Chli and R. Siegwart. "Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2012. DOI: 10.1109/icra.2012.6225147 (cit. on pp. 7, 58, 60).

[131] X. Ying and Z. Hu. "Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model". In: *Computer Vision - ECCV 2004*. Ed. by T. Pajdla and J. Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 442–455. ISBN: 978-3-540-24670-1. DOI: 10.1007/978-3-540-24670-1_34 (cit. on p. 42).

[132] Z. Zhang, H. Rebecq, C. Forster and D. Scaramuzza. "Benefit of large field-of-view cameras for visual odometry". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 801–808. DOI: 10.1109/ICRA.2016.7487210 (cit. on p. 38).

[133] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell and S. S. Srinivasa. "CHOMP: Covariant Hamiltonian optimization for motion planning". In: *The International Journal of Robotics Research* (2013). DOI: 10.1177/0278364913488805 (cit. on pp. 9, 120).

Curriculum vitae

Vladyslav Usenko
geboren am 29. Januar 1991
in Kiew, Ukraine

2014–2018	*Technische Universität München, Deutschland* Promotion am Lehrstuhl für Bildverarbeitung und Künstliche Intelligenz Betreut von Prof. Dr. Daniel Cremers
2011–2013	*Technische Universität München, Deutschland* Master of Science in Informatik Mit Auszeichnung bestanden
2007–2011	*Nationale Taras-Schewtschenko-Universität Kiew, Ukraine* Bachelor of Science in Informatik Mit Auszeichnung bestanden
2004–2007	*Naturwissenschaftliches Gymnasium No. 145, Kiew, Ukraine*
2000–2004	*Klovsky Gymnasium No. 77, Kiew, Ukraine*
1997–2000	*Grundschule No. 322, Kiew, Ukraine*