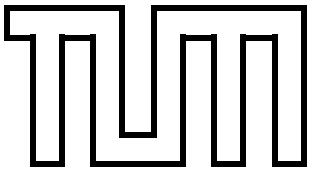


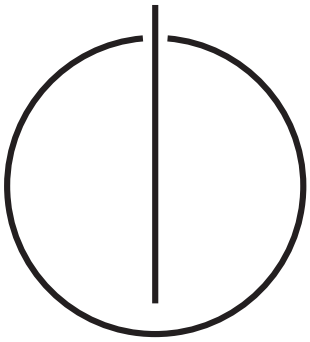
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

PhD Thesis

Large-Scale Direct SLAM and 3D Reconstruction in Real-Time

Jakob-Julian Engel



Technische Universität München
Fakultät für Informatik

Large-Scale Direct SLAM and 3D Reconstruction in Real-Time

Jakob-Julian Engel

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Nassir Navab
Prüfer der Dissertation:	1. Prof. Dr. Daniel Cremers
	2. Prof. Dr. Andrew Davison Imperial College London

Die Dissertation wurde am 11.10.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.03.2017 angenommen.

Abstract

This thesis explores direct formulations for real-time, incremental Structure from Motion / SLAM. Direct methods – in contrast to indirect methods – estimate geometry directly from the images, i.e., the raw sensor measurements, without intermediate abstraction, for instance in the form of keypoint matches. This has the major advantage that it does not require points to be recognizable on their own, thus allowing to use all information from the images including corners, edges, and weakly textured or repetitive image regions. Furthermore, is allows to employ a more finely grained (pixel wise) geometry representation.

The first part of the thesis proposes a novel large-scale direct monocular SLAM system (LSD-SLAM). LSD-SLAM employs a semi-dense formulation: Geometry is estimated in the form of smooth, semi-dense depth maps, which are obtained by filtering over many small-baseline stereo comparisons. In turn, the camera is tracked using direct image alignment, optimizing the relative pose of two frames directly on the raw images. Subsequent integration into a scale-drift aware pose-graph allows to perform large-scale mapping, including loop-closure detection and correction as well as relocalization. LSD-SLAM runs in real-time on a CPU and even on a modern smart phone. Furthermore, it is extended to other sensor modalities like stereo and omnidirectional cameras.

In the second part of the thesis, a novel direct and sparse formulation for monocular visual odometry (DSO) is proposed. It combines a direct model and error formulation with a sparse geometry representation, i.e., does not incorporate or enforce a geometry prior such as smoothness. This allows to jointly optimize all involved parameters, effectively performing the direct equivalent to full bundle adjustment. At the same time, as direct method, DSO employs a pixel-wise inverse depth representation and can use all image regions instead of only corners. Extensive experiments, comprising thousands of run sequences, show that the proposed direct and sparse formulation substantially outperforms the indirect approach, both in terms of accuracy as well as robustness.

Zusammenfassung

Diese Doktorarbeit untersucht direkte Ansätze für Structure from Motion und visuelles SLAM (Simultaneous Localization and Mapping). Direkte Methoden – im Gegensatz zum traditionellen indirekten Ansatz – berechnen 3D Geometrie und Kamerabewegung direkt auf den von der Kamera aufgenommenen Bildern, ohne diese zu diskreten Landmarken zu abstrahieren. Dies erlaubt es, alle Bildinformationen zu verwerten – insbesondere auch Kanten, leichte Schattierungen auf intensitätshomogenen Flächen, sowie repetitive Bildregionen. Desweiteren ermöglicht es der direkte Ansatz, Geometrie auf natürliche Weise in als präzise inverse Tiefenkarte zu repräsentieren.

Der erste Teil dieser Arbeit entwickelt eine direkte, semidichte SLAM Methode zur Rekonstruktion weitläufiger Gebiete ("Large-scale direct monocular SLAM", LSD-SLAM). LSD-SLAM repräsentiert Geometrie in der Form von semidichten Tiefenkarten unter Benutzung eines smoothness-Priors. Tiefenwerte werden durch die effiziente, probabilistische Fusion großer Mengen von Stereovergleichen berechnet (Filterung), während die Kamerabewegung mittels direct image alignment, also direkt anhand der Bilder berechnet wird. Die daraus resultierenden relativen Positionsinformationen zwischen Paaren von Bildern werden in einem zweiten Schritt in Form eines skalierungs-adaptiven Pose-Graphs gemeinsam optimiert, um global konsistente Positionen für jedes Kamerabild zu erhalten. Desweiteren integriert LSD-SLAM eine Komponente zur Kamera-Relokalisierung, sowie zum Erkennen großer Schleifenschlüsse. LSD-SLAM wurde als open-source code veröffentlicht, und operiert in Echtzeit auf der CPU, ohne eine GPGPU zu benötigen. Im weiteren Verlauf der Arbeit wird LSD-SLAM auf andere Sensormodalitäten erweitert, insbesondere wird LSD-SLAM für Stereokameras sowie omnidirektionale Kameras angepasst.

Der zweite Teil dieser Arbeit formuliert eine neuartige direkte und sparse Formulierung des Strukture from Motion Problems in Form einer neu entwickelten Methode für Visuelle Odometrie (DSO). DSO kombiniert den direkten Ansatz – insbesondere das dahinterstehende probabilistische Modell – mit einer sparsen Geometrie-Repräsentation. Insbesondere bedeutet dies, das DSO, im Gegensatz zu anderen direkten Ansätzen, keine Glattheitsannahme macht, sondern einzelne Punkte als unabhängig betrachtet (gegeben die Kamerapositionen). Dies erlaubt es, alle Modellparameter gemeinsam zu optimieren, und entspricht damit dem direkten Äquivalent zu (indirektem) Bündelausgleich. Da DSO einen direkten Ansatz verfolgt, kann es jedoch weiterhin alle Bildregionen, und damit alle Bildinformationen, benutzen – und ist nicht, wie der klassische indirekte Ansatz, auf eindeutig wiedererkennba-

re Landmarken beschränkt. Anhand von ausgiebigen Experimenten wird demonstriert, dass der vorgeschlagene direkte und sparse Ansatz deutlich höhere Genauigkeit ermöglicht, und gleichzeitig wesentlich robuster ist als der klassische indirekte Ansatz.

Acknowledgements

My colleagues & friends at the TUM cvpr lab. Jürgen, for getting me exited about quadrocopters, guiding me in my first research projects (despite my "advice resistant" tendency), and giving me the best introduction to research I could have wished for! Sabine and Quirin, for somehow managing to make things go smoothly in spite of my tardiness and lacking organizational skills. Daniel, for building and leading an awesome research group!

My flatmates from Munich's best WG. The place I probably did the largest part of this work (mostly between 1 and 5 in the morning). Regi, Franzi, Anna, and all my friends in Munich. Sheryl. Thanks for keeping the workaholic nerd in me at bay. It's been an awesome time!

My family. Thanks for your love and support, in spite of my occasional communication laziness. I wouldn't be where I am without you!

Contents

Part I

Introduction & Motivation

Chapter 1

Introduction

We humans perceive the world with our eyes. While we possess and use other senses such as touch and hearing, the sheer portion of the human brain devoted to visual processing – about 30%, compared to 8% for touch and 3% for hearing – demonstrates both the importance as well as the complexity of the ability to understand the 3D world around us from 2D projections observed by our eyes. While vision includes a large number of tasks including recognizing different people and objects, a fundamental component is the ability to perceive the 3D structure of the world, allowing us to explore unknown environments, drive a car along the road, or interact with physical objects in our presence.

As artificial devices – autonomous cars, quadrocopters, full-sized robots or even virtual and augmented reality systems – start to interact with, or adapt to the 3D world around us, they need the ability to perceive, reconstruct and ultimately understand it in a similar manner: A car that drives itself needs to know where it is, and it needs to recognize and avoid obstacles, both dynamic and static. To convincingly display a virtual object standing on a real-world table, both the pose of the observer, as well as the pose and shape of the table need to be known.

Researchers in computer vision, mathematics and robotics have thus spent decades on the task of reconstructing the 3D world – geometry and camera motion – from 2D images. It is commonly called Simultaneous Localization and Mapping (SLAM) or Structure and Motion (SaM).

1.1 A Brief History

In this section we give a brief history of the origins of today's SLAM / SaM formulations, summarizing important milestones and paradigm-shifts from the last decades. A more comprehensive analysis of the current state of the art will be given in Section 1.4.

Approaches for computing 3D structure from 2D images date back more than 100 years, long before the advent of digital photography or even computers. Very

early work includes that of Kruppa in 1913 [72], where he formulated an analytic approach to compute the relative pose of two images from 5 manually labelled point-correspondences. In the following decades, both the number of points and the number of images increased, and practical methods for solving the resulting mathematical systems were developed. The general term "Bundle Adjustment" – jointly optimizing a "Bundle" of rays – appears as early as 1976 in the work of D. C. Brown [20], still operating on manually labelled point correspondences in analogue images.

With the advent of digital imaging, the initial step of manually selecting and matching suitable points (landmarks) was replaced by automatic feature detection. First approaches to find suitable points – keypoints – include that of Förstner and Gülch [42] in 1987 and that of Harris and Stephens [54] in 1988. In the following years, other approaches such as FAST corners [97], which are significantly faster to compute, have been developed. Initially, selected keypoints were tracked by minimizing the photometric error between small patches around them. This is commonly known as the Kanade-Lucas-Tomasi feature tracker (KLT), which was first proposed in 1991 [117]. Only later, the local optimization approach was replaced by a global search in an abstracted descriptor space such as SIFT in 1999 [80], SURF in 2006 [18] or ORB in 2011 [98]. This effectively allows to solve the matching problem globally by approximating it with a nearest-neighbour search, replacing gradient-based local optimization.

After detecting and matching keypoints as a first step, *indirect* methods then proceed by estimating 3D geometry – camera motion and keypoint positions – from the found 2D correspondences. First real-time capable, incremental methods were based on Kalman filtering, i.e., accumulating information about the world as joint Gaussian distribution on all involved parameters. Early examples for such filtering-based algorithms include the work of Jin et al. [59], and the work of Davison et al. [31]. With PTAM (Parallel Tracking and Mapping), Klein et al. showed that real-time SLAM can also be formulated by representing the world in the form of a sparse, bipartite graph of keypoints and keyframes, which is optimized in the background using non-linear optimization (Bundle Adjustment).

Simultaneously to the appearance of keypoint detectors, *direct* and *semi-direct* formulations for structure and motion have been proposed. In contrast to the indirect approach (which separates correspondence estimation from geometric optimization), direct methods optimize 3D geometry on the raw intensity images without intermediate abstraction. An early example for direct and dense geometry estimation is the work of Matthies et al. in 1988 [81], which proposes a method to estimate dense depth from a (calibrated) sequence of images, using pixel-wise filtering interleaved with spatial smoothing. The work of Hanna in 1991 [53] proposes a direct formulation for estimating dense depth as well as the camera motion from a monocu-

lar image sequence, minimizing a photometric error. A sparse and direct monocular SLAM system capable of running in real-time was presented by Jin et al. in 2002 [58], which optimizes sparse patch positions & normals as well as camera poses for a sequence of images, using a photometric error formulation.

1.2 Classification of SLAM Methods

This section defines three ways to classify visual SLAM / SaM methods: direct vs. indirect, dense vs. sparse, and optimization-based vs. filtering-based. These classifications are intentionally kept independent of the used sensor modalities. In fact they apply in the context of different sensor combinations and modalities, including monocular, stereo, visual-inertial, and RGB-D.

1.2.1 Direct vs. Indirect

Underlying almost all formulations is a probabilistic model that takes noisy measurements $\mathbf{Y}$ as input and computes an estimator $\mathbf{X}$ for the unknown, hidden model parameters (3D world model & camera motion). Typically a Maximum Likelihood approach is used, which finds the model parameters that maximize the probability of obtaining the actual measurements, i.e.,

$$\mathbf{X}^* := \operatorname*{argmax}_{\mathbf{X}} P(\mathbf{Y}|\mathbf{X}). \tag{1.1}$$

Indirect methods then proceed in two steps: First, the raw sensor measurements are pre-processed to generate an intermediate representation, solving part of the overall problem, such as establishing correspondences. Second, the computed intermediate values are interpreted as noisy measurements $\mathbf{Y}$ in a probabilistic model to estimate geometry and camera motion. Note that the first step is typically approached by extracting and matching a sparse set of keypoints – however other options exist, like establishing correspondences in the form of dense, regularized optical flow. This also includes methods that extract and match parametric representations of other geometric primitives than only points, such as line- or curve-segments.

Direct methods skip the pre-processing step and directly use the actual sensor values – light received from a certain direction over a certain time period – as measurements $\mathbf{Y}$ in a probabilistic model.

In the case of passive vision, the direct approach thus optimizes a *photometric error*, since the sensor provides photometric measurements. Indirect methods on the other hand optimize a *geometric error*, since the pre-computed values – point-positions or flow-vectors – are geometric quantities. Note that for other sensor modalities like depth cameras or laser scanners (which directly measure geometric

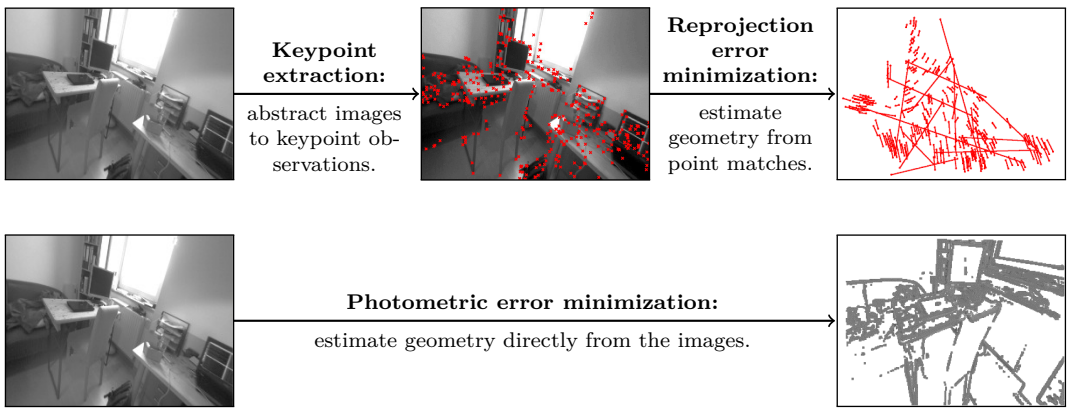


Figure 1.1: **Direct vs. indirect.** The *indirect* approach (top) splits the overall task – estimating geometry and camera motion from images – into two sequential steps, keypoint detection and matching, and geometric optimization on the computed point correspondences. The *direct* approach (bottom) in turn skips this intermediate representation, and directly optimizes geometry and camera motion on the raw intensity images.

quantities) direct formulations may also optimize a geometric error. Figure 1.1 conceptually visualizes the difference between the direct and the indirect approach.

1.2.2 Dense vs. Sparse

Sparse methods use and reconstruct only a selected set of independent points (traditionally corners), whereas *dense* methods attempt to use and reconstruct all pixels in the 2D image domain. Intermediate approaches (*semi-dense*) refrain from reconstructing the complete surface, but still aim at using and reconstructing a (largely connected & well-constrained) subset of it.

Apart from the extent of the used image region however, a more fundamental – and consequential – difference lies in the addition of a geometry prior. In the sparse formulation, there is no notion of neighborhood, and geometry parameters (keypoint positions) are conditionally independent given the camera poses & intrinsics[1]. Dense (or semi-dense) approaches on the other hand exploit the connectedness of the used image region to formulate a geometry prior, typically favoring smoothness. In fact, such a prior is necessarily required to make a dense world model observable from passive vision alone. In general, this prior is formulated directly in the form of an additional log-likelihood energy term. Figure 1.2 shows two inverse depth maps, a semi-dense one created by LSD-SLAM, and a sparse one created by DSO.

[1]Note that even though early filtering-based methods such as [31] kept track of point-point-correlations, these originated from marginalized camera poses, not from the model itself.

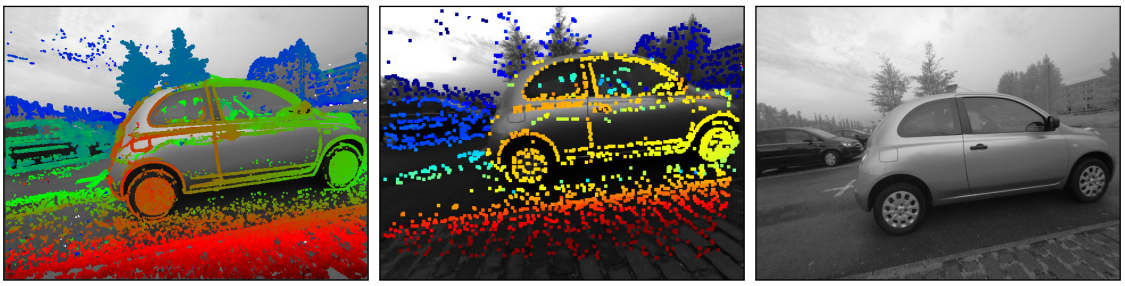


Figure 1.2: **Dense vs. sparse.** Left: Semi-dense, spatially regularized depth map created by LSD-SLAM. The regularizer assumes neighboring pixels to have similar depth, thereby correlating them in the model. Middle: Sparse inverse depth map created by DSO. No smoothness prior is included in the model, thus all points are conditionally independent given the camera poses. Right: Original image.

1.2.3 Filtering vs. Optimization

The fundamental difference between filtering-based and optimization-based approaches lies in the way the internal state (i.e., the world model) is represented.

Filtering-based methods estimate and continuously update a joint probability distribution over all relevant parameters. New measurements are used to update this distribution (reducing uncertainty), while time progression adds new parameters with large initial uncertainty or increases the uncertainty of existing ones. This approach is commonly known as Kalman filtering. This representation allows to easily marginalize old state variables, thus filtering-based methods typically marginalize old states, and only keep the current camera pose in the state vector.

Recent filtering-based methods such as the multi-state constrained Kalman filter (MSCKF) include a sliding window of past camera poses and intrinsic calibration parameters, but do not include geometry parameters (point positions). Instead, these are only added once they leave the field of view, and then immediately marginalized.

Optimization-based methods on the other hand keep information in the form of a non-linear energy function, which is continuously optimized in the background. To facilitate this, they aggressively drop available information and only keep a small subset of frames (keyframes). This has the advantage of lower computational complexity in terms of the number of points ($\mathcal{O}(m^3 + m^2 n)$ instead of $\mathcal{O}\Big((m+n)^3\Big)$ for filtering, where m is the number of frames and n the number of points). Further, it allows linearizations to be re-evaluated after better estimates are available. On the other hand, it reduces the number of actual observations that can be incorporated into the system.

In [110], Strasdat et al. draw the conclusion that filtering has a better accuracy

Figure 1.3: **Augmented Reality.** Left: external view of an augmented reality application running on a smartphone. Right: screenshot recorded on the phone, with a virtual car driving over a real-world surface. The system is discussed in Chapter 5, was developed as joint work with Thomas Schöps, and is published in [104, 11]

.

per unit of computation trade-off for small systems, (i.e., towards the low end of available compute budged), whereas optimization-based methods achieve a better trade-off for larger problems, using significantly more points.

1.3 Applications for SLAM

Real-time visual SLAM and visual odometry have many practical applications, which are becoming increasingly important for current technological developments. The most prominent areas are

Visual and Augmented Reality (VR / AR). The camera pose and the geometry of the scene are required to correctly render virtual objects into the image, and allow them to interact (e.g., collide with or disappear behind) real-world objects. At the same time, wearable devices – such as a headsets or smart phones – impose severe restrictions on the cost, size, weight, and power consumption of the used sensors. This leads to passive vision becoming an important sensor modality – both in a monocular or stereo set-up, and typically combined with an IMU. Figure 1.3 shows an example of an AR system developed as part of this thesis running on a modern smart phone, it will be presented in Chapter 5.

Robotics. Examples include autonomous quadrocopters, driver-less cars, and robot vacuum cleaners. Visual SLAM is used to estimate the robot's position with respect to the environment and to navigate without colliding with other objects. While in some cases other sensor modalities (such as laser scanners or RGB-D cameras) can be used, using passive vision is a good – and sometimes the only – option for resource-constrained systems, due to restrictions on per-unit production cost or on size / weight / power consumption. Figure 1.4 shows a number of examples of

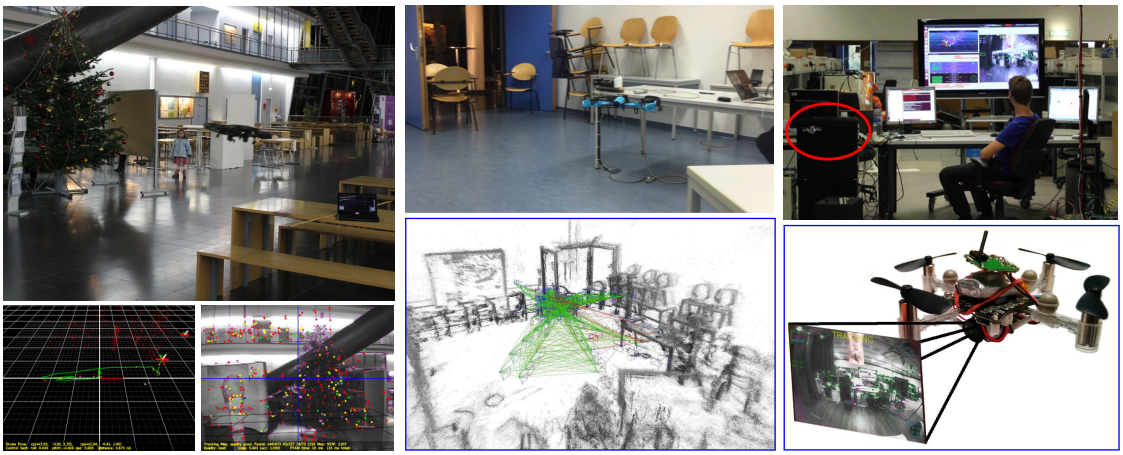


Figure 1.4: **Visual SLAM for robotics.** Left: Parrot AR.Drone flying a pre-defined path autonomously, using PTAM and a front-facing camera. This line of work is published in [38, 6–8]. Middle: Parrot Bebop, using LSD-SLAM to autonomously explore and reconstruct an unknown environment, autonomously avoiding obstacles. This line of work was joint work with Lukas von Stumberg, and is published in [113]. Right: a 25g Nano-Copter equipped with a miniature camera and analogue video transition, using computer vision to fly a simple figure. This line of work was joint work with Oliver Dunkley, and is published in [34, 2].

robotic systems developed in conjunction with or prior to this thesis, using visual SLAM for navigation, 3D reconstruction and obstacle avoidance. Note that we have chosen to not include the respective publications in this cumulative thesis in order to keep it thematically consistent, and since some of the work forms part of other thesises. Table 1.1 in Section 1.5 gives an exhaustive list of all publications.

1.4 State of the Art

In this section we list relevant state-of-the art SLAM and VO systems, all of which have been proposed shortly before or during the preparation of this thesis. We only list methods that are based on passive vision, i.e., using one or more passive cameras as supposed to RBG-D sensors.

DTAM: Dense Tracking and Mapping [88]. DTAM was one of the first dense and direct SLAM methods that operate on a single monocular camera. Dense depth maps are estimated for keyframes by accumulating information from many small-baseline images in a perspective cost volume, from which a dense depth map is extracted using a variational optimization approach. The camera pose in turn is tracked by direct image alignment, minimizing the photometric difference between the observed image, and a predicted view rendered from the dense world model.

DTAM runs in real-time on a GPU, and was later extended to fuse individual depth maps into a volumetric world model [87]. It was preceded by the work of Stühmer et al., [112], which uses a similar approach for dense depth estimation but assumes given camera poses.

SVO: Semi-Direct Visual Odometry [41]. SVO is a sparse hybrid between the direct and indirect approach. Point depths are initially estimated using a direct formulation, optimizing the photometric error between the observed image and the reference patch for the given point. New frames in turn are aligned using direct image alignment. Subsequently, the authors "relax" the epipolar constraint, effectively fixing the established correspondences and converting the formulation to an indirect one, in order to optimize the system jointly. Effectively, this means that a direct formulation is used to obtain robust and outlier-free initializations for the underlying indirect model. SVO is very computationally efficient, running in real-time on a CPU and even on computationally constrained embedded processors. The approach was later extended to include line segment features in addition to point features, and to operate on stereo-, visual-inertial, and omnidirectional data.

ORB-SLAM: A Versatile and Accurate Monocular SLAM System [86]. ORB-SLAM is a classical sparse and indirect SLAM system. The map is optimized in the background using traditional bundle adjustment, while new frames are tracked in real-time using model-based tracking. The system is very well engineered, and characterized by its robustness, accuracy and flexibility. Furthermore, it includes relocalization and loop-closure detection, and can handle large maps, using a double-window optimization strategy. It was later extended to stereo- and RGB-D SLAM, and runs in real-time on laptop CPU.

MSCKF: Multi-State Constraint Kalman Filter [85]. The MSCKF is a sparse, filtering-based, indirect odometry method for a monocular camera combined with an IMU (visual-inertial odometry). Where most previous approaches treated visual-inertial odometry as sensor-fusion problem (modeling vision as black-box 5DoF/6DoF sensor), the MSCKF tightly couples both modalities, thereby optimally exploiting their complementary nature. In fact, it is described as vision-aided inertial odometry, rather than inertial-aided visual odometry. As the name suggests, the MSCKF is formulated as extended Kalman filter, keeping as state a sliding window of recent camera frames. 3D points are triangulated and added as observation only once they leave the field of view of the camera, thus they are never included in the state vector[2]. The MSCKF was extended to a rolling-shutter camera model in

[2] in later publications, features visible for a long time are added to the state vector in order to facilitate tracking through long periods in which the camera does not move.

[77], and is very computationally efficient – running in real-time on a modern smart phone.

OKVIS: Keyframe-based visual-inertial odometry using non-linear optimization [75]. OKVIS is a sparse, indirect visual-inertial odometry method. It optimizes a non-linear error function combining visual terms (reprojection error) and inertial terms over a sliding window of old frames and points. Parameters that fall out of this window are permanently marginalized, using the Schur complement. OKVIS is the optimization-based complement to the MSCKF, and runs in real-time on a CPU.

1.5 Contribution and Outline

This thesis develops novel *direct* solutions for real-time SLAM, also called structure and motion. In contrast to the state of the art, the developed direct methods do not rely on keypoint detection and matching for establishing correspondences across the input images, but rather optimize geometry and camera motion directly on the raw sensor measurements, i.e., the intensity images.

This cumulative thesis comprises 7 full-length publications [1, 3–5, 9–11], which are the result of joint work with Thomas Schöps, David Caruso, Vladyslav Usenko, Jörg Stückler, Jürgen Sturm, Prof. Vladlen Koltun and Prof. Daniel Cremers. Five of these works [1, 4, 5, 9, 11] were published in highly ranked, peer-reviewed international conferences and journals. [3] has been published as open-access pre-print, and has been submitted to *IEEE Transactions on Pattern Recognition and Machine Intelligence*. [10] has been published as open-access pre-print. Table 1.1 shows a complete summary of all works published as part of, or in conjunction with this thesis. It also lists second-author publications, as well as publications originating from different (but closely related) projects, which are not included as part of this cumulative thesis.

In this thesis, two complete systems representing two complementary approaches (a sparse, direct and a semi-dense, direct one) are developed. Furthermore, a novel dataset for the evaluation of monocular visual odometry systems, including novel photometric calibration schemes and benchmark metrics, is presented.

1.5.1 Large-Scale, Direct Monocular SLAM (LSD-SLAM)

LSD-SLAM is a novel approach to perform fully direct SLAM in real-time on a CPU. As direct method, it does not rely on classical features (keypoints), and thus can use – and reconstruct – all image regions that carry information, including edges

Camera-Based Navigation of a Low-Cost Quadrocopter. Jakob Engel, Jürgen Sturm, and Daniel Cremers; In: *IROS 2012* [7].
Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing. Jakob Engel, Jürgen Sturm, and Daniel Cremers; In: *ICRA 2012 Workshop (ViCoMoR)* [6].
Semi-Dense Visual Odometry for a Monocular Camera. Jakob Engel, Jürgen Sturm, and Daniel Cremers; In: *ICCV 2013* [9] (Chapter 3).
Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera. Jakob Engel, Jürgen Sturm, and Daniel Cremers; In: *RAS 2014, 62.11 (2014), 1646–1656* [8].
Visual-Inertial Navigation for a Camera-Equipped 25g Nano-Quadrotor. Oliver Dunkley, Jakob Engel, Jürgen Sturm, and Daniel Cremers; In: *IROS 2014 Workshop (Aerial Open Source)* [2].
LSD-SLAM: Large-Scale Direct Monocular SLAM. Jakob Engel, Thomas Schöps, and Daniel Cremers; In: *ECCV 2014* [4] (Chapter 4).
Semi-Dense Visual Odometry for AR on a Smartphone. Thomas Schöps, Jakob Engel, and Daniel Cremers; In: *ISMAR 2014* [11] (Chapter 5).
Large-Scale Direct SLAM for Omnidirectional Cameras. David Caruso, Jakob Engel, and Daniel Cremers; In: *IROS 2015* [1] (Chapter 7).
Large-Scale Direct SLAM with Stereo Cameras. Jakob Engel, Jörg Stückler, and Daniel Cremers; In: *IROS 2015* [5] (Chapter 6).
Reconstructing Street-Scenes in Real-Time From a Driving Car. Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers; In: *3DV 2015* [13].
Direct Visual-Inertial Odometry with Stereo Cameras. Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers; In: *ICRA 2016* [12].
Direct Sparse Odometry. Jakob Engel, Vladlen Koltun, and Daniel Cremers; In: *arXiv 2016 (submitted to TPAMI)* [3] (Chapter 8).
A Photometrically Calibrated Benchmark For Monocular Visual Odometry. Jakob Engel, Vladyslav Usenko, and Daniel Cremers; In: *arXiv 2016* [10] (Chapter 9).

Table 1.1: **Full Publication Summary.** Complete list of publications, ordered chronologically. For publications that form part of this cumulative thesis, we list the respective chapter. Publications not included in this thesis are listed in gray.

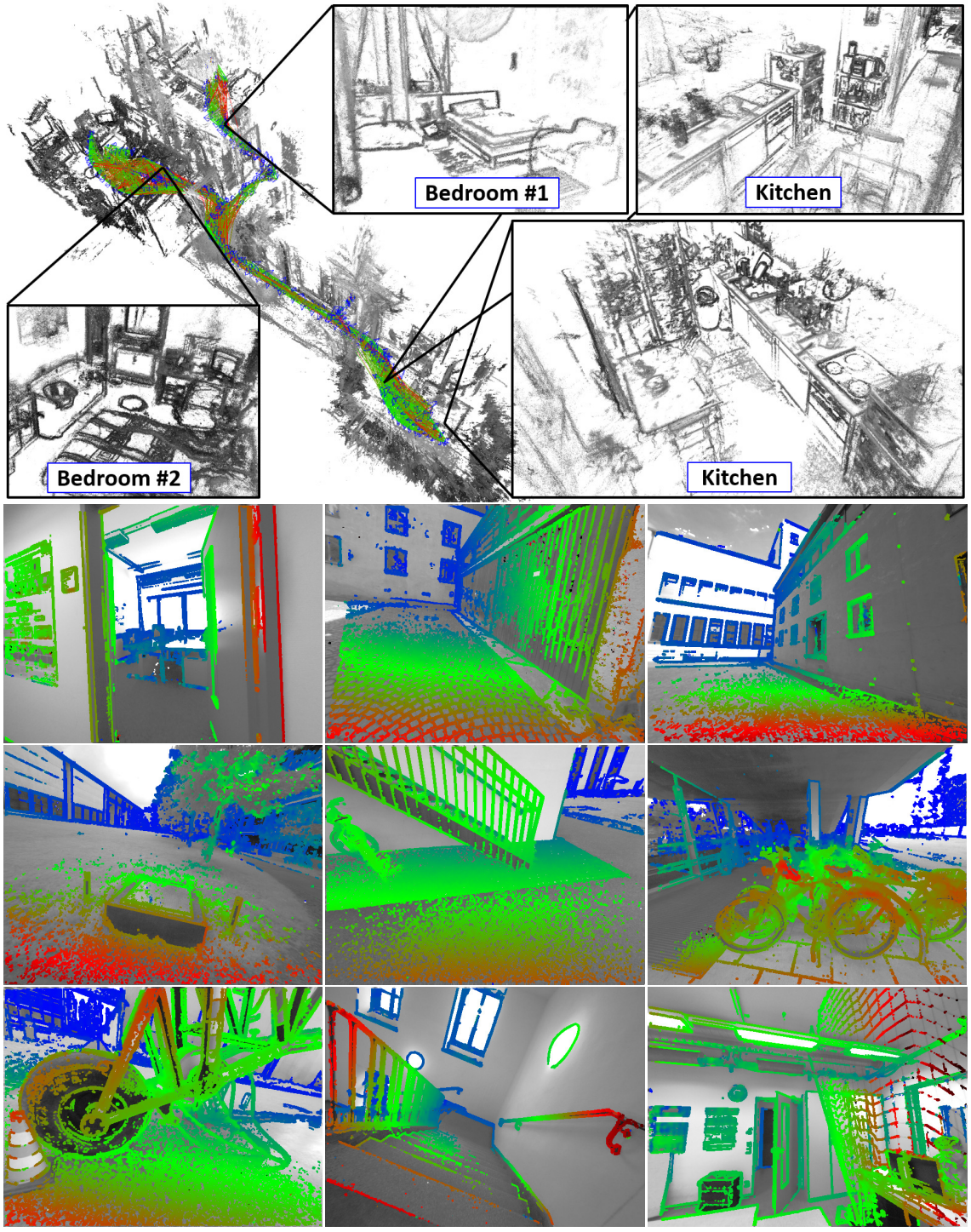


Figure 1.5: **LSD-SLAM.** The top shows an example of a large, semi-dense reconstruction of an entire flat, created with LSD-SLAM – including loop-closure and global map optimization. The bottom shows a number of color-coded semi-dense depth maps from different scenes of the TUM monoVO dataset. For further details, see Chapter 4.

and densely textured surfaces. This increases robustness to strong motion blur and sparsely textured environments compared to traditional indirect approaches.

Geometry is represented in the form of semi-dense inverse depth maps for selected keyframes, containing depth values for all pixels with sufficient intensity gradient. For each pixel, a one-dimensional extended Kalman filter is used to estimate depth from many small-baseline stereo comparisons to other frames, which is interleaved with spatial regularization (edge-preserving smoothing) and outlier removal. New frames are tracked in real-time using direct SE(3) image alignment to the closest keyframe. For global map optimization, keyframes are aligned towards each other using direct Sim(3) image alignment, and the global map optimized in a pose-graph. Figure 1.5 shows an example of some semi-dense depth maps and reconstructed semi-dense 3D models.

The real-time odometry component of LSD-SLAM is described in [9] (Chapter 3), the integration into a full SLAM system in [4] (Chapter 4). A smartphone-based implementation, as well as integration with a basic augmented reality engine is presented in [11] (Chapter 5). Furthermore, LSD-SLAM is extended to stereo in [5] (Chapter 6), and to an omnidirectional camera (with a field of view above 180°) in [1] (Chapter 7). [12] – which does not form part of this thesis – describes an approach to tightly integrate an IMU, further increasing robustness and accuracy of the system significantly.

1.5.2 Direct Sparse Odometry (DSO)

DSO is a monocular visual odometry system, which, like LSD-SLAM, does not rely on recognizable point-features or corners, but rather optimizes all parameters on the raw intensity values. In contrast to LSD-SLAM however, DSO uses a sparse formulation, i.e., does not spatially regularize geometry, and instead sparsely samples points from across many different keyframes. This allows to optimize all involved parameters – geometry, camera poses and camera intrinsics – in a joint, consistent Gauss-Newton framework. Note that the direct formulation still allows DSO to samples points from across all image regions that carry information, including points on edges and weak intensity variations on mostly white walls. The resulting reconstructions are similar in completeness to LSD-SLAM, however – when run with real-time settings – much more sparse; an example is shown in Figure 1.6. DSO takes advantage of a complete photometric calibration, including the camera response function, pixel-wise light attenuation factors (vignetting) and auto-exposure compensation.

DSO was published in [3] (Chapter 8), and significantly surpasses all previous monocular SLAM or VO methods (indirect approaches as well as direct approaches) in tracking accuracy and robustness. This is demonstrated on three publicly available datasets, comprising several hours of video, in a large variety of real-world scenes.

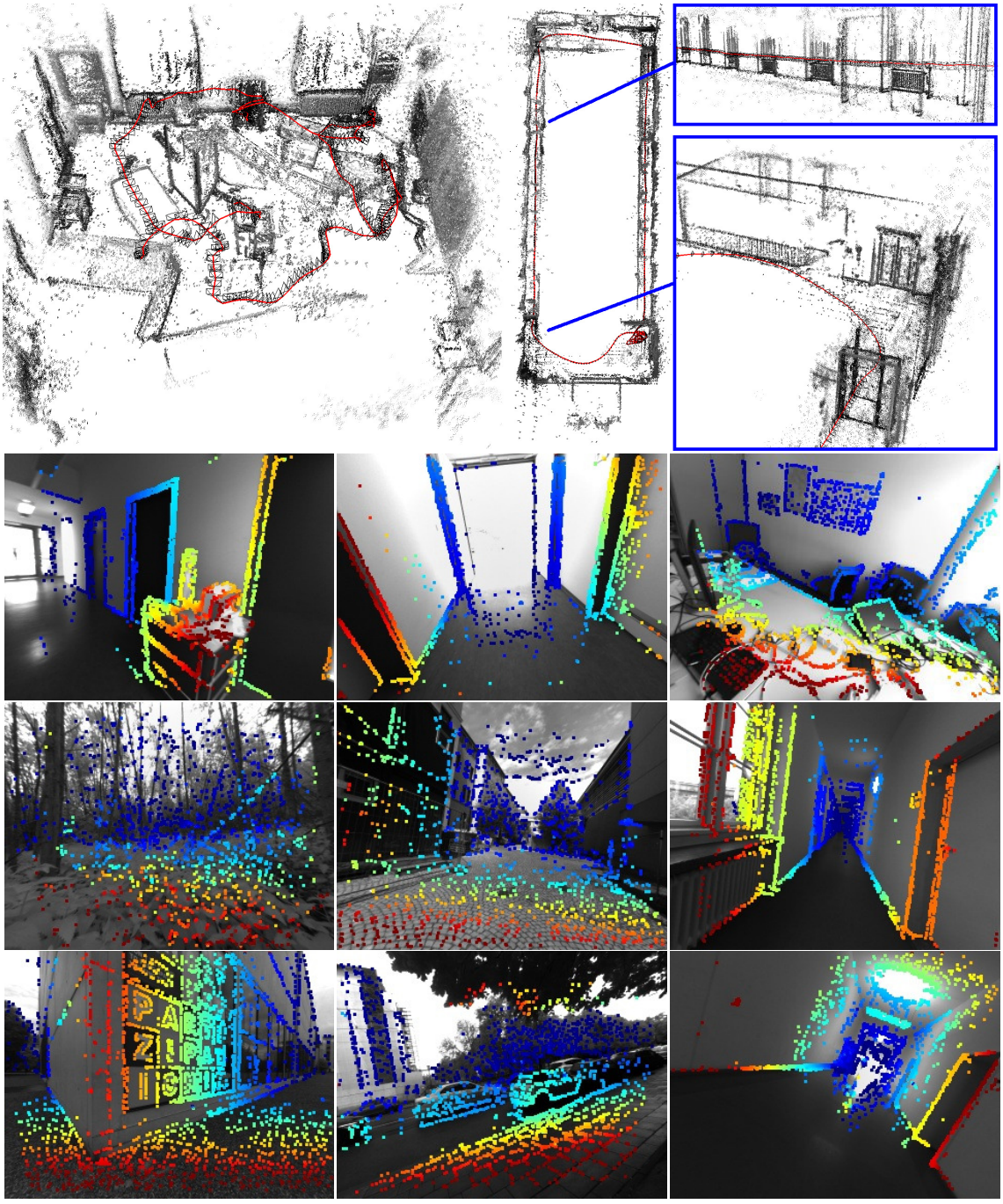


Figure 1.6: **DSO.** The top shows an example of two long trajectories, tracked with DSO. Even without loop-closure, the accumulated tracking drift is so small, that it is not visible in the figures. The bottom shows a number of color-coded sparse depth maps from different scenes of the TUM monoVO dataset. For further details, see Chapter 8.

1.5.3 TUM MonoVO Benchmark Dataset

The TUM monoVO benchmark dataset comprises 50 photometrically calibrated monocular sequences with a total length of 105 minutes (190'000 frames), taken in a large variety of different environments under natural, hand-held motion. We propose a novel evaluation metric to evaluate the tracking accuracy of a visual odometry method via the accumulated drift after a large loop-closure. Furthermore, a novel approach for photometric camera calibration (including pixel response function and a non-parametric vignetting map) is proposed. The dataset was published in [10] (Chapter 9).

We thoroughly evaluate and contrast DSO with a state-of-the-art indirect monocular SLAM system (ORB-SLAM), highlighting the effects of photometric noise vs. geometric noise on both approaches. Furthermore, we analyze the effect of various system design choices, such as field of view of the camera, image resolution, number of points used, and the number of keyframes taken – in total, the results shown in [3] and [10] are based on more than 100 million tracked video frames.

Figure 1.7: **The TUM monoVO dataset.** A single frame from each of the 50 sequences. Note the wide variety of covered environments, including narrow indoor corridors as well as wide outdoor scenes. It is further described in Chapter 9.

Chapter 2

Fundamentals

In this chapter we summarize the fundamental mathematical tools and computer vision concepts used throughout the thesis.

2.1 Camera Calibration

In order to use the data acquired by any sensor, we need to know how this data relates to the real world. Generally speaking, the more – and the more precise – the available knowledge about the measurement process, the better it can be exploited. This section discusses two parts of the camera calibration process: Section 2.1.1 describes commonly used models for geometric calibration, which define the function that maps a 3D point onto a pixel position in the image. Section 2.1.2 describes photometric calibration, which defines how scene irradiance is transformed into pixel intensity values. While photometric calibration is often ignored as it provides little benefit for keypoint-based techniques, it can greatly improve the results of direct methods.

2.1.1 Geometric Camera Calibration

The intrinsic camera calibration is a function $\pi\colon \mathbb{R}^3 \to \Omega$ which maps a 3D point in the cameras coordinate frame to a 2D point in the image. We will only consider *central* camera models, that is, models where all light rays intersect in one point, the camera center. In this case, the projected position of a point only depends on its direction from the camera center, and not its distance. Note that real-world cameras are never strictly central – in order to capture sufficient light, they have a non-zero area in which light is passed through to the sensor (the aperture), causing points that are too close or too far away from the camera to be blurred. In practice however, the aperture is sufficiently small, such that an approximation as central system is valid.

Pinhole Projection Model

For a given point $\mathbf{x} := (x, y, z)^T \in \mathbb{R}^3$ in 3D, we first define the general perspective projection function $\pi \colon \mathbb{R}^3 \to \mathbb{R}^3$ as

$$\pi(\mathbf{x}) = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} := \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2.1}$$

We then define the *Camera Matrix* $\mathbf{K}$ as

$$\mathbf{K} := \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where f_x, f_y corresponds to the camera's focal length, and c_x, c_y to the cameras principal point. The projected pixel position (u, v) of a point $\mathbf{x}$ can then be calculated as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\pi\left(\mathbf{x}\right). \tag{2.3}$$

This camera model has the nice property that it is linear, and hence maps straight lines in 3D to straight lines in the image, greatly simplifying many computations like stereo matching. For a point with known depth, the projection is straight-forward to invert (*re-projection*):

$$\mathbf{x} = \pi^{-1}\left(\begin{bmatrix} u \\ v \end{bmatrix}, z\right) = \mathbf{K}^{-1} z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{2.4}$$

where z is the known depth.

Non-linear Distortion

Real-world lenses – in particular lenses with a wide field of view – do not behave according to the raw pinhole projection model. A common approach to solve this is to add a non-linear function $\gamma \colon \mathbb{R}^2 \to \mathbb{R}^2$ (*distortion*) to the projected point coordinates, to obtain the final projected point position:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{K}\gamma\left(\pi\left(\mathbf{x}\right)\right). \tag{2.5}$$

Note that we apply the distortion function to the projected point coordinates before multiplying with the camera matrix $\mathbf{K}$ – for readability, we omit the homogeneous

coordinate where it is 1. It is important to keep in mind that the true projection function is ultimately defined by the used lens and the placement of the lens with respect to the image sensor. In practice, there is a number of common ways to approximate this function in closed form, here we will the two most commonly used variants.

FOV model. This model is specifically designed for fisheye lenses and was first proposed by Fevernay and Faugeras [33]. It assumes that the distance of a given pixel in the image to the principal point is proportional to the angle between the optical axis and the ray connecting the optical center with the 3D point. This model assumes radially symmetric distortion, i.e.,

$$\gamma_{\mathrm{F}}\left(\begin{bmatrix} u_u \\ v_u \end{bmatrix}\right) := \begin{bmatrix} u_d \\ v_d \end{bmatrix} := \frac{R(r_u)}{r_u}\begin{bmatrix} u_u \\ v_u \end{bmatrix}, \tag{2.6}$$

where $r_u := \sqrt{u_u^2 + v_u^2}$. The function R is characterized by only one parameter ω

$$R(r_u) := \frac{1}{\omega}\arctan\left(2r_u \tan\left(\frac{\omega}{2}\right)\right) \tag{2.7}$$

A useful property of this model is the existence of a closed-form inverse

$$\gamma_{\mathrm{F}}^{-1}\left(\begin{bmatrix} u_d \\ v_d \end{bmatrix}\right) = \begin{bmatrix} u_u \\ v_u \end{bmatrix} = \frac{R^{-1}(r_d)}{r_d}\begin{bmatrix} u_d \\ v_d \end{bmatrix}, \tag{2.8}$$

with $r_d := \sqrt{u_d^2 + v_d^2}$ and

$$R^{-1}(r_d) := \frac{\tan(r_d\omega)}{2\tan\frac{\omega}{2}}. \tag{2.9}$$

Radio-Tangential Model. A widely used approach is to approximate the distortion using a polynomial of degree n – higher degree approximations have more expressive power, but are more difficult to calibrate and may become numerically unstable. Here we give as example the full model used as default in OpenCV, which has a total of 8 parameters (6 radial coefficients, κ_1 to κ_6 and 2 tangential coefficients, ρ_1 and ρ_2). Higher-degree coefficients can be set to zero, effectively reducing the degree of the used approximation. This model is well suited to remove distortion from consumer-grade cameras, which can be afflicted by significant tangential distortion. It is however not well suited to model fisheye lenses with large distortion, and large field of view (120° and more). The distortion function is given by

$$\gamma_{\mathrm{T}}\left(\begin{bmatrix} u_u \\ v_u \end{bmatrix}\right) := \begin{bmatrix} u_u \frac{1+\kappa_1 r_u^2+\kappa_2 r_u^4+\kappa_3 r_u^6}{1+\kappa_4 r_u^2+\kappa_5 r_u^4+\kappa_6 r_u^6} + 2\rho_1 u_u v_u + \rho_2(r_u^2+2u_u^2) \\ v_u \frac{1+\kappa_1 r_u^2+\kappa_2 r_u^4+\kappa_3 r_u^6}{1+\kappa_4 r_u^2+\kappa_5 r_u^4+\kappa_6 r_u^6} + \rho_1(r_u^2+2v_u^2) + 2\rho_2 u_u v_u \end{bmatrix}, \tag{2.10}$$

where $r_u := \sqrt{u_u^2 + v_u^2}$ is defined as before. Note that this distortion model has no closed-form inverse. Furthermore, in contrast to the FOV model it does not assume strict radial symmetry. While in practice, non-radial distortion is uncommon in high-quality cameras, it may be caused e.g. by the optical axis of the lens not being exactly perpendicular to the sensor.

Unified Omnidirectional Projection Model

One of the fundamental limitations of the pinhole model – even when applying non-linear distortion – is, that it can only model lenses with a field of view below 180°. To model optical systems with a larger field of view, we will use the unified omnidirectional model, which is characterized by spherical projection followed by a pinhole projection. That way, it can model optical systems with arbitrarily large field of view; as long as they can be approximated as a central system. A given point $\mathbf{x} := (x, y, z)^T \in \mathbb{R}^3$ in 3D is first projected onto the unit sphere by normalizing it to length one. Afterwards, perspective projection using a camera center shifted by $-\xi$ is applied:

$$\pi_{\mathrm{U}}(\mathbf{x}) = \begin{bmatrix} \frac{x}{z+\xi\|\mathbf{x}\|} \\ \frac{y}{z+\xi\|\mathbf{x}\|} \\ 1 \end{bmatrix}. \tag{2.11}$$

One of the benefits of this formulation is again the existence of a closed-form inverse, given by

$$\pi_{\mathrm{U}}^{-1}\left(\begin{bmatrix} u \\ v \end{bmatrix}, \|\mathbf{x}\|\right) = \|\mathbf{x}\| \left(\frac{\xi + \sqrt{1 + (1-\xi^2)\left(u^2+v^2\right)}}{u^2+v^2+1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix} \right). \tag{2.12}$$

Afterwards, multiplication with the camera matrix $\mathbf{K}$ as well as non-linear distortion are applied, the full projection function is hence again given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{K}\gamma\left(\pi_{\mathrm{U}}\left(\mathbf{x}\right)\right). \tag{2.13}$$

Note that pinhole projection can be seen as a special case of the unified model with $\xi = 0$.

Image Rectification

Image rectification describes the process of warping an image to fit a different projection model. It is commonly used to remove non-linear distortion in a pre-processing step, such that sub-sequent computations can use a raw pinhole or unified projection model.

Figure 2.1: **Image rectification.** Left: original, radially distorted image – note how straight lines in the real world project to curves in the image. Middle: full rectified image, using Equation (2.14), note that the undistorted image domain is not square – however, straight lines in the real world project to straight lines in the image. The right image shows the rectified image, cropped to the maximal square region that is well-defined.

Given an original image $I\colon \Omega \to \mathbb{R}$ and the corresponding calibrated projection function $\pi\colon \mathbb{R}^3 \to \Omega$, we can warp the image to compute a synthetic image $I'\colon \Omega' \to \mathbb{R}$ following a new projection function $\pi'\colon \mathbb{R}^3 \to \Omega'$ as

$$I'(\mathbf{x}) = I\left(\pi\left(\pi'^{-1}(\mathbf{x}, 1)\right)\right). \tag{2.14}$$

The new projection function π', as well as the domain of the synthetic (rectified) image Ω' can be defined arbitrarily. Generally, Ω' is chosen such that it preserves a maximal area of the original image, while not including any regions that are not covered in the original image. Figure 2.1 shows an example of image rectification.

Note that rectifying images changes the spatial sampling of the underlying continuous image function, which can introduce significant blur or aliasing effects. It is thus advisable to choose the new projection function π' such that it minimizes the introduced distortions.

2.1.2 Photometric Camera Calibration

A fundamental assumption for direct methods is the *brightness constancy assumption*, stating that points in 3D have the same color when observed in different images. In practice, it is not the pixel value that is constant over time, but the emitted energy per unit area per unit solid angle $\left[\frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{sr}}\right]$, called *radiance* of a given 3D surface point. Photometric (or radiometric) calibration describes the process of calibrating the relation between scene radiance and the intensity value measured by the sensor. In the context of multi-view stereo – in order to match geometry across different images, taken with the same (or equivalent) camera – only a relative calibration (i.e., up to scale) is required.

While there is a large number of physical phenomena affecting the measurement process, in this thesis we only consider the most influential factors: exposure time,

pixel response, and lens vignetting. The overall photometric sensor model is thus given by

$$I(\mathbf{x}) = G\Big(tV(\mathbf{x})B(\mathbf{x})\Big), \tag{2.15}$$

where $B\colon \Omega \to \mathbb{R}$ is the scene radiance, $V\colon \Omega \to [0 \dots 1]$ the pixel-wise light attenuation caused by lens vignetting, t the exposure time, and $G\colon \mathbb{R} \to \{0 \dots 255\}$ the response function. Since we operate on monochromatic images, white-balancing is not included in the model. The simplest way to incorporate a photometric calibration into photometric stereo is to apply the inverse of (2.15) to compute B for all images as a very first step in the algorithmic pipeline, and use this instead of the original image I.

A straight-forward and easy to reproduce approach to photometrically calibrate a camera will be proposed in Chapter 9, and incorporated into a direct monocular visual odometry method (DSO) in Chapter 8.

2.2 3D Geometry

This thesis is about estimating motion in, and geometry of the three-dimensional world, hence representing position and orientation of points and objects in 3D is of particular importance. In this section we will introduce basic 3D geometric calculations and representations, which will be used throughout the remainder of the thesis.

We define a 3D coordinate transformation from frame A to frame B in terms of its relative translation $t_A^B \in \mathbb{R}^3$ and rotation $R_A^B \in \mathrm{SO}(3)$, which form the transformation matrix $T_A^B \in \mathrm{SE}(3)$. Here, SE(3) stands for the special Euclidean group defined as

$$\mathrm{SE}(3) := \left\{ \left[\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right] | R \in \mathrm{SO}(3), t \in \mathbb{R} \right\}. \tag{2.16}$$

The rotation R is from the special orthogonal group

$$\mathrm{SO}(3) := \left\{ R \in \mathbb{R}^{3\times 3} : RR^T = \mathbf{1} \wedge \ \det(R) = +1 \right\}. \tag{2.17}$$

A 3D point $\mathbf{x}_A$ in frame A can then be transformed into frame B by multiplication with the respective transformation matrix

$$\tilde{\mathbf{x}}_B = T_A^B \tilde{\mathbf{x}}_A \tag{2.18}$$

where the tilde $\tilde{\cdot}$ represents a points homogeneous representation. Using this representation, transformation matrices can be concatenated by simple multiplication

$$T_A^C = T_B^C T_A^B \tag{2.19}$$

and inverted

$$T_B^A = (T_A^B)^{-1} = \begin{pmatrix} R^T & -R^T t \\ 0 & 1 \end{pmatrix}. \tag{2.20}$$

While this matrix representation is often convenient to work with, it is an over-parameterization of the underlying group (using 3 parameters for translation and 9 for rotation), which only has 6 degrees of freedom (3 for translation and 3 for rotation). In the following we will give some alternative representations for 3D poses which are used when less parameters and less, or more simple intrinsic constraints are required, as is the case for optimization.

2.2.1 Quaternions

Quaternions represent 3D rotation as a point on the 4D unit sphere. They are the most compact *singularity-free* rotation representation. The direct SLAM and visual odometry systems presented in this thesis use quaternions for internal state representation. Given a quaternion $q = (x, y, z, w)^T \in \mathbb{S}^3$, the corresponding rotation matrix can be computed as

$$R = \begin{pmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 \end{pmatrix}. \tag{2.21}$$

A minimal representation can be derived from this by simply dropping w – it can be recovered from the unit length constraint as $w = \sqrt{1-x^2-y^2-z^2}$.

2.2.2 Euler Angles

Euler angles represent 3D orientation as three angles, denoting three sequential rotations around certain axis. There are many possibilities how these axis and the order in which they are rotated can be defined, which in practice renders this representation ambiguous. As any minimal representation of SO(3), Euler angles suffer from singularities. A particular problem for Euler Angles is, that in certain configurations they lose a degree of freedom ("Gimbal Lock"), i.e., they locally loose the ability to be perturbed in all three dimensions and the Jacobian will lose one rank – making this representation unsuited for optimization.

One of the most common use cases is in the context of aircraft navigation, where the three angles are termed *roll* (rotation around the front-to-back axis of the aircraft), *pitch* (rotation around the left-to-right axis of the aircraft) and *yaw* (rotation around the vertical axis). As in this case both roll and pitch are typically small (except for extreme flight maneuvers), Gimbal locks do not occur.

2.2.3 Lie Representation

A particularly elegant way to minimally represent 3D poses can be derived using Lie group theory. A Lie group is a group that is also a smooth manifold, where the group operations (multiplication and inversion) are smooth maps. There are many Lie Groups, the most important ones in the context of this thesis being the *Special Euclidean Group* as defined in Eq. 2.16, the *Special Orthogonal Group* as defined in Eq. 2.17, and the group of *3D Similarity Transforms* Sim(3), defined as

$$\mathrm{Sim}(3) := \left\{ \left[\begin{array}{ccc|c} & sR & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \middle| R \in \mathrm{SO}(3), t \in \mathbb{R}^3, s \in \mathbb{R}^+ \right\}. \tag{2.22}$$

Note that in the literature (e.g. [35]) Sim(3) can also be defined as

$$\mathrm{Sim}(3) = \left\{ \left[\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & s^{-1} \end{array}\right] | R \in \mathrm{SO}(3), t \in \mathbb{R}^3 s \in \mathbb{R}^+ \right\}, \tag{2.23}$$

which leads to different choice of generators (see below), as well as different results for the Adjoint representation, the logarithmic and the exponential map. Essentially, (2.22) first applies scaling, then rotation and translation, whereas (2.23) first applies rotation and translation, and then scaling.

Each *Lie Group* gives rise to a corresponding *Lie Algebra*, which defines the structure of the tangent space around the identity. It is given by all linear combinations of its generator matrices, which are the derivatives with respect to elementary rotation, translation and scaling evaluated around the identity. They are given by

$$G_{r_x} := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_{r_y} := \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_{r_z} := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G_{t_x} := \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_{t_y} := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_{t_z} := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G_s := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

We can now define the Lie algebras corresponding to the introduced Lie groups SO(3), SE(3), and Sim(3) as

$$\mathfrak{so}(3) := \left\{ \omega_1 G_{r_x} + \omega_2 G_{r_y} + \omega_3 G_{r_z} | (\omega_1, \omega_2, \omega_3) \in \mathbb{R}^3 \right\}, \tag{2.24}$$

$$\mathfrak{se}(3) := \left\{ A + v_1 G_{t_x} + v_2 G_{t_y} + v_3 G_{t_z} | A \in \mathfrak{so}(3), (v_1, v_2, v_3) \in \mathbb{R}^3 \right\}, \tag{2.25}$$

$$\mathfrak{sim}(3) := \left\{ B + \lambda G_s | B \in \mathfrak{se}(3), \lambda \in \mathbb{R} \right\}. \tag{2.26}$$

Elements can be mapped from a Lie algebra to the respective Lie group using the matrix exponential, and vice-versa from the Lie group to the Lie algebra using the matrix logarithm. For convenience, we will denote elements of a Lie algebra directly with vectors containing the respective generator matrix coefficients when convenient, omitting the in the literature often used hat ($\hat{\cdot}$) and vee ($\cdot^\vee$) operators.

The resulting representation is ideally suited for differential quantities on a Lie group (such as uncertainty or derivatives), as

- it is a minimal representation with an exact, surjective mapping $\exp\colon \mathfrak{se}(3) \to \mathrm{SE}(3)$.
- the Adjoint can be used to linearly and exactly map elements from one tangent space to another (see Section 2.2.3). This also means that all tangent-spaces have the same (linear) structure.
- it is defined in terms of the derivatives with respect to the underlying motion types, and hence intuitive to understand, and to use in the context of gradient-based optimization.

In the remainder of this section, we give a number of useful derivations and definitions for the Adjoint, the Jacobian, and uncertainty representation on Lie groups. For brevity and ease of notation, we only consider SE(3) – however, all given derivations can be generalized to SO(3) and Sim(3).

The Adjoint

We will follow the approach by Eade [35] and directly define the Adjoint by its most important property: The Adjoint is a function that "moves" elements between different tangent spaces of a Lie group. Given a group element T and an algebra element ξ with

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(3) \qquad \xi = \begin{pmatrix} u \\ \omega \end{pmatrix} \in \mathfrak{se}(3). \tag{2.27}$$

The Adjoint $\mathrm{Adj}_T \in \mathbb{R}^{6\times 6}$ is a linear function that "moves" ξ from the right tangent space of T to the left tangent space of T, i.e.,

$$T \cdot e^{\widehat{\xi}} = e^{\widehat{\mathrm{Adj}_T \cdot \xi}} \cdot T. \tag{2.28}$$

We can obtain a closed-form expression for the Adjoint by using that $Ae^X A^{-1} = e^{AXA^{-1}}$ for any invertible matrix A, i.e.,

$$e^{\widehat{\mathrm{Adj}_T \cdot \xi}} = T \cdot e^{\widehat{\xi}} \cdot T^{-1} \tag{2.29}$$

$$= e^{T \cdot \widehat{\xi} \cdot T^{-1}} \tag{2.30}$$

$$\widehat{\mathrm{Adj}_T \cdot \xi} = T \cdot \widehat{\xi} \cdot T^{-1} \tag{2.31}$$

which can be solved for Adj_T:

$$\mathrm{Adj}_T = \begin{bmatrix} R & t_\times R \\ 0_{3\times 3} & R \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{2.32}$$

The Adjoint representation for SO(3) and Sim(3) can for example be found in [35, 79, 107].

Jacobians

The definition of a Lie algebra element ξ as coefficient vector of the generator matrices G_1 to G_6 and $\frac{\partial e^{Gx}}{\partial x}\Big|_{x=0} = G$ directly implies that

$$\frac{\partial e^{\widehat{\xi}}}{\partial \xi_i}\bigg|_{\xi=0} = G_i \qquad \text{for} \qquad i = 1 \dots 6. \tag{2.33}$$

In the context of 3D reconstruction and SLAM, almost always derivatives of functions that involve rotating and translating points are required. Let

$$\tilde{\mathbf{y}}(\xi, \mathbf{x}) := e^{\widehat{\xi}}\tilde{\mathbf{x}} \tag{2.34}$$

define the 3D transformation of a point (note that for ease of notation, we include the homogeneous coordinate – in practice it is always one, and the respective derivative always zero). Since $\tilde{\mathbf{y}}$ is linear in $e^{\widehat{\xi}}$ and $\tilde{\mathbf{x}}$, the derivative with respect to ξ around the identity can again be obtained by the generators

$$\frac{\partial \tilde{\mathbf{y}}(\xi, \mathbf{x})}{\partial \xi_i}\bigg|_{\xi=0} = \frac{\partial e^{\widehat{\xi}}}{\partial \xi_i}\bigg|_{\xi=0} \tilde{\mathbf{x}} = G_i\tilde{\mathbf{x}} \qquad \text{for} \qquad i = 1 \dots 6. \tag{2.35}$$

The Adjoint can then be used to derive functions involving pose concatenations: Given a function

$$\tilde{\mathbf{y}'}(\xi, \mathbf{x}) := Te^{\widehat{\xi}}\tilde{\mathbf{x}}. \tag{2.36}$$

We can use (2.28) to first "move" ξ all the way to the left

$$\tilde{\mathbf{y}'}(\xi, \mathbf{x}) = e^{\widehat{\mathrm{Adj}_T\xi}}T\tilde{\mathbf{x}}, \tag{2.37}$$

and then use (2.35) to compute the derivative as

$$\frac{\partial \tilde{\mathbf{y}'}(\xi, \mathbf{x})}{\partial \xi}\bigg|_{\xi=0} = \frac{\partial e^{\widehat{\xi'}}T\tilde{\mathbf{x}}}{\partial \xi'}\bigg|_{\xi'=0} \frac{\partial \mathrm{Adj}_T\xi}{\partial \xi}\bigg|_{\xi=0} = \left[G_1T\tilde{\mathbf{x}}|\dots|G_6T\tilde{\mathbf{x}}\right] \cdot \mathrm{Adj}_T, \tag{2.38}$$

where we use $[\cdot|\dots|\cdot]$ to denote the full Jacobian obtained from writing the partial derivative vectors from (2.35) in matrix form. Note that these derivations only hold if the exponential map is evaluated **around zero**.

Uncertainty representation on poses

In a Gaussian framework, uncertainties on a Lie-group element $T \in \mathrm{SE}(3)$ are best represented in the respective tangent space, i.e.,

$$T =: e^{\widehat{\epsilon}}T_0 \qquad \text{with} \qquad \epsilon \sim \mathcal{N}(0, \Sigma_\epsilon). \tag{2.39}$$

Using the above Jacobian derivations, the Adjoint can then be used to propagate uncertainties through pose concatenations as

$$T' = T_a T T_b =: e^{\widehat{\epsilon'}} \underbrace{T_a T_0 T_b}_{=T'_0} \quad \text{with} \quad \epsilon' \sim \mathcal{N}(0, \underbrace{\mathrm{Adj}_{T_a} \Sigma_\epsilon \mathrm{Adj}_{T_a}^T}_{=\Sigma_{\epsilon'}}), \tag{2.40}$$

and through pose inversion as

$$T' = T^{-1} =: e^{\widehat{\epsilon'}} \underbrace{T_0^{-1}}_{=T'_0} \quad \text{with} \quad \epsilon' \sim \mathcal{N}(0, \underbrace{\mathrm{Adj}_{T^{-1}} \Sigma_\epsilon \mathrm{Adj}_{T^{-1}}^T}_{=\Sigma_{\epsilon'}}). \tag{2.41}$$

Note that due to the linearity of the Adjoint, propagating uncertainties in this way is exact, i.e., does not involve (additional) linearizations. In particular, this implies that the accuracy of the linearizations required to "fit" the true distribution to a Gaussian does not depend on the specific tangent space chosen to represent it in. Equivalently, we will show in the following chapter that when optimizing over poses in this multiplicative representation, the computed updates will not depend on the specific tangent space they are computed in. In practice however, choosing an unsuitable tangent space can cause numerical issues.

2.3 Non-Linear Least-Squares Optimization

Many computer vision problems can be formulated as finding the minimizer of an energy function E. This is often complemented with other algorithmical procedures, for instance to determine suitable (key-)points and (key-)frames, to identify outliers, and to find sufficiently good initializations for the parameters optimized over. All methods developed in this thesis have at their core an energy function which is minimized to find the desired model parameters, such as camera poses or geometry parameters. It generally has a non-linear least-squares form, i.e., can be written as

$$E(\mathbf{x}) = \sum_i r_i^2(\mathbf{x}), \tag{2.42}$$

where $\mathbf{x}$ is the function argument, and the r_i are non-linear scalar functions which can have arbitrary form. Least-squares minimization problems arise naturally as log-likelihood of *probabilistically independent* and *Gaussian distributed* measurements. In most practical cases, the residual functions are non-convex, making the overall energy function E non-convex as well. They are often written as stacked residual vector, i.e.,

$$E(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|_2^2, \quad \text{with} \quad \mathbf{r} = [r_1, \ldots, r_n]^T. \tag{2.43}$$

In Section 2.3.1 we formulate the Gauss-Newton algorithm for minimizing least-square energy functions, as well as well-known extensions such as Levenberg-Marquad, iterative reweighing, and the Schur complement trick. Section 2.3.2 describes an elegant way to optimize over non-Euclidean manifolds, such as Lie groups.

2.3.1 Gauss-Newton Optimization

The Gauss-Newton algorithm iteratively computes and solves a quadratic approximation to E by linearizing the residual function $\mathbf{r}$, using a first-order Taylor expansion

$$\begin{aligned} E(\mathbf{x}_0 + \boldsymbol{\delta}) &= \|\mathbf{r}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2 && (2.44)\\ &\approx \|\mathbf{r}_0 + \mathbf{J}_\mathbf{r}\boldsymbol{\delta}\|_2^2 && (2.45)\\ &= \mathbf{r}_0^T\mathbf{r}_0 + 2\boldsymbol{\delta}^T \underbrace{\mathbf{J}_\mathbf{r}^T\mathbf{r}_0}_{=:\mathbf{b}} + \boldsymbol{\delta}^T \underbrace{\mathbf{J}_\mathbf{r}^T\mathbf{J}_\mathbf{r}}_{=:\mathbf{H}}\boldsymbol{\delta}, && (2.46) \end{aligned}$$

with

$$\mathbf{J}_\mathbf{r} = \left.\frac{d\mathbf{r}(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}_0} \quad \text{and} \quad \mathbf{r}_0 = \mathbf{r}(\mathbf{x}_0). \tag{2.47}$$

Setting the derivative of the resulting quadratic function to zero gives

$$\mathbf{H}\boldsymbol{\delta} + \mathbf{b} = 0, \tag{2.48}$$

which can be solved for the increment $\boldsymbol{\delta}$ as

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{b} = -\underbrace{(\mathbf{J}_\mathbf{r}^T\mathbf{J}_\mathbf{r})^{-1}\mathbf{J}_\mathbf{r}^T}_{=\mathbf{J}_\mathbf{r}^\dagger}\mathbf{r}_0, \tag{2.49}$$

where $\mathbf{J}_\mathbf{r}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{J}_\mathbf{r}$. If the system is non-degenerate (i.e., all parameters are fully observable), it has full rank and there exists a unique solution. The computed increment is added to the evaluation point to give the new estimate

$$\mathbf{x} \leftarrow \mathbf{x}_0 + \boldsymbol{\delta}. \tag{2.50}$$

This process is repeated until convergence.

Note that while the Gauss-Newton method computes and solves a *quadratic* approximation to E using a *linear* approximation to $\mathbf{r}$, this only *approximates* the second-order Taylor expansion of E. The exact second-order expansion is given by

$$\frac{dE(\mathbf{x})}{d\mathbf{x}} = 2\mathbf{J}_\mathbf{r}^T\mathbf{r}_0 = 2\mathbf{b} \tag{2.51}$$

$$\left[\frac{d^2E(\mathbf{x})}{d\mathbf{x}^2}\right]_{jk} = 2\sum_i \left(\frac{\partial r_i}{\partial \mathbf{x}_j}\frac{\partial r_i}{\partial \mathbf{x}_k} + r_i\frac{\partial^2 r_i}{\partial \mathbf{x}_j \partial \mathbf{x}_k}\right) \tag{2.52}$$

$$\approx 2\sum_i \left(\frac{\partial r_i}{\partial \mathbf{x}_j}\frac{\partial r_i}{\partial \mathbf{x}_k}\right) = 2\left[\mathbf{H}\right]_{jk}, \tag{2.53}$$

where $[\cdot]_{ik}$ denotes the corresponding matrix entry. In practice, this is a good approximation if $\frac{\partial r_i}{\partial \mathbf{x}_j}\frac{\partial r_i}{\partial \mathbf{x}_k} \gg r_i\frac{\partial^2 r_i}{\partial \mathbf{x}_j \partial \mathbf{x}_k}$, which is the case if the residuals r_i are small or close to linear.

In many cases it is desirable to weight the residuals differently, i.e., minimize a function of the form

$$E(\mathbf{x}) = \sum_i w_i r_i^2(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|_\mathbf{W}^2, \tag{2.54}$$

where w_i are *constant*, *non-negative* weights, and $\mathbf{W}$ is the respective diagonal weight matrix. The w_i can either directly be incorporated into the residual functions as $r_i'(\mathbf{x}) := \sqrt{w_i}r_i(\mathbf{x})$, or explicitly kept as separate weights, which leads to $\mathbf{H} = \mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r}$ and $\mathbf{b} = \mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{r}_0$.

Levenberg-Marquad

The Gauss-Newton algorithm offers quadratic convergence rates close to the minimum, however it can become unstable for poorly initialized parameters. In such

cases, simple gradient descent is more reliable, since – for sufficiently small step-size and under certain conditions – it is guaranteed to converge to a local minimum. The Levenberg-Marquad algorithm is designed to resolve this by adaptively "interpolating" between Gauss-Newton and gradient descent steps via a dampening parameter λ. The updates are then computed as

$$\boldsymbol{\delta} = -(\mathbf{H} + \lambda \operatorname{diag}(\mathbf{H}))^{-1}\mathbf{b}, \tag{2.55}$$

where $\operatorname{diag}(\mathbf{H})$ is the matrix containing only the diagonal entries of $\mathbf{H}$. If λ is small, $\mathbf{H}$ dominates and the computed step will be close to the original Gauss-Newton step from (2.49). In turn, if λ is large, $\lambda \operatorname{diag}(\mathbf{H})$ dominates and the computed step will be close to a gradient descent step, with step-size $\frac{1}{\lambda}$, and scaled with the curvature along each dimension to avoid slow convergence along dimensions with low gradient. The full Levenberg-Marquad algorithm then proceeds as follows:

Algorithm 1: Levenberg Marquad Algorithm

Initialize λ, and $\mathbf{x}^{(0)}$;
do
  $\mathbf{J}_\mathbf{r} := \left.\frac{d\mathbf{r}(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}^{(n)}}$ and $\mathbf{r}_0 := \mathbf{r}(\mathbf{x}^{(n)})$;
  $\mathbf{H} := \mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r}$ and $\mathbf{b} = \mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{r}_0$;
  $\boldsymbol{\delta} := -\Big(\mathbf{H} + \lambda\operatorname{diag}(\mathbf{H})\Big)^{-1}\mathbf{b}$;
  if $E(\mathbf{x}^{(n)} + \boldsymbol{\delta}) < E(\mathbf{x}^{(n)})$ **then**
    $\mathbf{x}^{(n+1)} \leftarrow \mathbf{x}^{(n)} + \boldsymbol{\delta}$;
    Decrease λ;
  else
    $\mathbf{x}^{(n+1)} \leftarrow \mathbf{x}^{(n)}$;
    Increase λ;
  end
while $\|\boldsymbol{\delta}\| > \epsilon$;

The best strategy for increasing, decreasing, and initializing λ depends on the shape and structure of E, on the accuracy of the initialization $\mathbf{x}^{(0)}$, and on the relative computation cost of evaluating E, computing $\mathbf{H}$, $\mathbf{b}$, and solving for $\boldsymbol{\delta}$. When minimizing a photometric error, we found that using $\lambda_{\text{inc}} = 5\lambda$, $\lambda_{\text{dec}} = \frac{1}{2}\lambda$, and $\lambda_0 = 0.01$ give a good trade-off in terms of convergence time.

The Schur Complement Trick

When optimizing over a large number of variables n (e.g., several camera poses and thousands of point positions), solving the $n \times n$ linear equation system $\mathbf{H}\mathbf{x} = \mathbf{b}$ becomes the computationally most expensive part of GN / LM optimization. In

practice, E often has a specific sparsity structure that can be exploited: The Schur complement trick can be applied if there exists a variable partitioning $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]^T$, with $n_1 := \dim(\mathbf{x}_1) \ll \dim(\mathbf{x}_2) =: n_2$, such that all components of $\mathbf{x}_2$ are conditionally independent given $\mathbf{x}_1$. In other words, this means that residuals may depend on any components from $\mathbf{x}_1$, but only on a single component from $\mathbf{x}_2$. Note that each component can be a low-dimensional cluster of variables, such as a 3D point or a scalar inverse depth value. We can write the $(n_1 + n_2) \times (n_1 + n_2)$ linear system as

$$\begin{pmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}, \tag{2.56}$$

where $\mathbf{H}_{22}$ is a (block-) diagonal matrix. The Woodbury matrix identity can then be used to show that

$$\Big(\mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{-1}\mathbf{H}_{21}\Big)\mathbf{x}_1 = \Big(\mathbf{b}_1 - \mathbf{H}_{12}\mathbf{H}_{22}^{-1}\mathbf{b}_2\Big), \tag{2.57}$$

which is a $n_1 \times n_1$ linear system and thus much faster to solve. In turn, $\mathbf{H}_{22}^{-1}$ is easy to invert since it is a (block-) diagonal matrix. Afterwards, the solution for $\mathbf{x}_2$ can be recovered by resubstitution, using

$$\mathbf{H}_{21}\mathbf{x}_1 + \mathbf{H}_{22}\mathbf{x}_2 = \mathbf{b}_2. \tag{2.58}$$

The Schur complement trick greatly speeds up *sparse* SLAM methods, where $\mathbf{x}_1$ contains the camera poses and $\mathbf{x}_2$ the (conditionally independent) geometry parameters. In practice, each GN / LM step then scales with $\mathcal{O}(n_1^2(n_1 + n_2))$, instead of a naïve implementation which scales with $\mathcal{O}((n_1 + n_2)^3)$. The Schur complement trick will be used in Chapter 8 in the context of Direct Sparse Odometry (DSO).

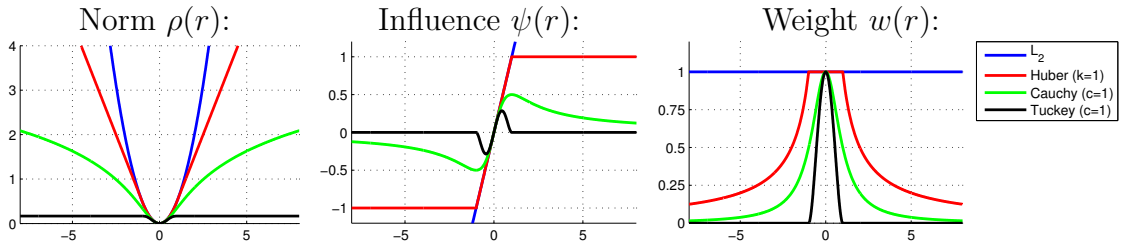
Iteratively Reweighed Least-Squares

A least-squared (L_2) error formulation corresponds to Gaussian distributed residuals. However, in most practical applications, residuals are not Gaussian distributed. Most importantly, there will be outliers with large residual values – in a least squares formulation, these have a strong impact on the solution, corrupting the system. To mitigate the influence of outliers and to approximate other residual distributions, the L_2 norm is replaced by robust error norms (M-estimators) $\rho\colon \mathbb{R} \to \mathbb{R}$ which are designed to reduce the influence of large residuals. The energy function becomes

$$E(\mathbf{x}) = \sum_i \rho\Big(r_i(\mathbf{x})\Big), \tag{2.59}$$

which is then minimized using *iteratively reweighed least-squares*. It can be derived as follows: First, we note that minimizing (2.59) corresponds to finding a zero-crossing of its derivative, i.e., finding $\mathbf{x}$ for which

$$0 = \frac{dE(\mathbf{x})}{d\mathbf{x}} = \sum_i \frac{d\rho(r_i)}{dr_i}\frac{dr_i(\mathbf{x})}{d\mathbf{x}}. \tag{2.60}$$



	Norm $\rho(r)$	Influence $\psi(r)$	Weight $w(r)$
L_2	$\frac{r^2}{2}$	r	1
Huber $\begin{cases}\lvert r\rvert < k \\ \lvert r\rvert \geq k\end{cases}$	$\begin{cases}\frac{r^2}{2} \\ k(\lvert r\rvert - \frac{k}{2})\end{cases}$	$\begin{cases}r \\ k\ \mathrm{sign}(r)\end{cases}$	$\begin{cases}1 \\ \frac{k}{\lvert r\rvert}\end{cases}$
Cauchy	$\frac{c^2}{2}\log\left(1+(\frac{r}{c})^2\right)$	$\frac{r}{1+(\frac{r}{c})^2}$	$\frac{1}{1+(\frac{r}{c})^2}$
Tuckey $\begin{cases}\lvert r\rvert < c \\ \lvert r\rvert \geq c\end{cases}$	$\begin{cases}\frac{c^2}{6}\left(1-\left(1-(\frac{r}{c})^2\right)^3\right) \\ \frac{c^2}{6}\end{cases}$	$\begin{cases}r\left(1-(\frac{r}{c})^2\right)^2 \\ 0\end{cases}$	$\begin{cases}\left(1-(\frac{r}{c})^2\right)^2 \\ 0\end{cases}$

Figure 2.2: **M-estimators.** Frequently used error norms ρ, their respective influence functions ψ, and weight functions w. The Huber norm has the advantage of being convex, but in turn still gives linear influence to outliers. For the Cauchy norm, the influence of outliers is bounded logarithmically, whereas for Tuckey it is explicitly set to zero, once a certain threshold is crossed.

we now define $\psi = \frac{d\rho(r)}{dr}$ (called the *influence function* of ρ), and expand the above expression to

$$0 = \frac{dE(\mathbf{x})}{d\mathbf{x}} = \sum_i \frac{\psi(r_i)}{r_i} r_i \frac{dr_i(\mathbf{x})}{d\mathbf{x}}, \tag{2.61}$$

where we define the *weight function* of ρ as $w(r) = \frac{\psi(r)}{r}$. This turns out to be exactly the system of equations obtained when minimizing

$$E'(\mathbf{x}) = \sum_i w\Big(r_i(\mathbf{x}_c)\Big) r_i^2(\mathbf{x}), \tag{2.62}$$

where $\mathbf{x}_c$ is held constant and set to the most recent estimate of $\mathbf{x}$, i.e., the current evaluation point. Essentially, this derivation transforms (2.59) into a weighted least-squares system by "fixing" the weights in each iteration. In practice, there exist many possible choices for ρ with different advantages and disadvantages – the most commonly used choices are summarized in Figure 2.2.

2.3.2 Gauss-Newton on non-Euclidean Manifolds

When optimizing over non-Euclidean spaces – such as rigid body poses – a challenge arises from the fact that in in many cases (including SO(3), SE(3), and Sim(3)), all parameterizations either have singularities (such as Euler angles and Lie algebra elements) or are over-parameterized (such as quaternions and rotation matrices). A common approach to solve this is to represent state estimates in a singularity-free, over-parameterized space, while computing $\boldsymbol{\delta}$-increments on a minimal parameterization, evaluated around a point that is far away from singularities. In the following, we will use Lie to denote the over-parameterized, singularity-free space (such as SO(3), SE(3), and Sim(3)), and $\mathfrak{lie}$ to denote the corresponding minimal representation (such as $\mathfrak{so}(3)$, $\mathfrak{se}(3)$, and $\mathfrak{sim}(3)$). We further define the

$$\boxplus : \mathfrak{lie} \times \text{Lie} \to \text{Lie} \tag{2.63}$$

operator, which applies an increment to a state estimate. When operating on Lie groups, it can be defined in a left-multiplicative formulation as

$$\xi \boxplus T := e^{\widehat{\xi}} T, \tag{2.64}$$

or in a right-multiplicative formulation as

$$\xi \boxplus T := T e^{\widehat{\xi}}. \tag{2.65}$$

Both formulations are *linearly* related by the Adjoint (see (2.28)), and therefore yield exactly equivalent update steps. Note however, that the computed intermediate values will be different for both formulations, which is important when interpreting the inverse Hessian $\mathbf{H}^{-1}$ as uncertainty.

Given a least-squares energy function $E\colon \text{Lie} \to \mathbb{R}$ and a current estimate $T^{(n)} \in \text{Lie}$, each GN / LM iteration then computes an increment $\boldsymbol{\delta} \in \mathfrak{lie}$ to a modified ("shifted") energy function $E^{(n)}\colon \mathfrak{lie}(3) \to \mathbb{R}$, defined as

$$E^{(n)}(\boldsymbol{\delta}) := E(\boldsymbol{\delta} \boxplus T^{(n)}), \tag{2.66}$$

which is evaluated around $\boldsymbol{\delta} = 0$. Correspondingly, the update step is replaced by a multiplicative update

$$T^{(n+1)} \leftarrow \boldsymbol{\delta} \boxplus T^{(n)}. \tag{2.67}$$

Note that in this formulation, the exponential map is always linearized around zero, which – in the case of Lie algebras or Euler angles – is far away from any singularities.

Part II

Own Publications

Chapter 3

Semi-Dense Visual Odometry for a Monocular Camera

Authors	Jakob Engel[1]	engelj@in.tum.de
	Jürgen Sturm[2]	sturmju@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] Google	

Publication **Semi-Dense Visual Odometry for a Monocular Camera**. J. Engel, J. Sturm, D. Cremers. In *Preceedings of IEEE International Conference on Computer Vision (ICCV), 2013*. Copyright 2013 IEEE. Reprinted with permission from IEEE. doi: 10.1109/ICCV.2013.183

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

Abstract We propose a fundamentally novel approach to real-time visual odometry for a monocular camera. It allows to benefit from the simplicity and accuracy of dense tracking – which does not depend on visual features – while running in real-time on a CPU. The key idea is to continuously estimate a semi-dense inverse depth map for the current frame, which in turn is used to track the motion of the camera using dense image alignment. More specifically, we estimate the depth of all pixels which have a non-negligible image gradient. Each estimate is represented as a Gaussian probability distribution over the inverse depth. We propagate this information over time, and update it with new measurements as new images arrive. In terms of tracking accuracy and computational speed, the proposed method compares favorably to both state-of-the-art dense and feature-based visual odometry and SLAM algorithms. As our method runs in real-time on a CPU, it is of large practical value for robotics and augmented reality applications.

3.1 Towards Dense Monocular Visual Odometry

Tracking a hand-held camera and recovering the three-dimensional structure of the environment in real-time is among the most prominent challenges in computer vision. In the last years, dense approaches to these challenges have become increasingly popular: Instead of operating solely on visual feature positions, they reconstruct and track on the whole image using a surface-based map and thereby are fundamentally different from feature-based approaches. Yet, these methods are to date either not real-time capable on standard CPUs [88, 112, 122] or require direct depth measurements from the sensor [66], making them unsuitable for many practical applications.

In this paper, we propose a novel semi-dense visual odometry approach for a monocular camera, which combines the accuracy and robustness of dense approaches with the efficiency of feature-based methods. Further, it computes highly accurate semi-dense depth maps from the monocular images, providing rich information about the 3D structure of the environment. We use the term visual odometry as supposed to SLAM, as – for simplicity – we deliberately maintain only information about the currently visible scene, instead of building a global world-model.

3.1.1 Related Work

Feature-based monocular SLAM. In all feature-based methods (such as [31, 69]), tracking and mapping consists of two separate steps: First, discrete feature observations (i.e., their locations in the image) are extracted and matched to each other. Second, the camera and the full feature poses are calculated from a set of such observations – disregarding the images themselves. While this preliminary abstrac-

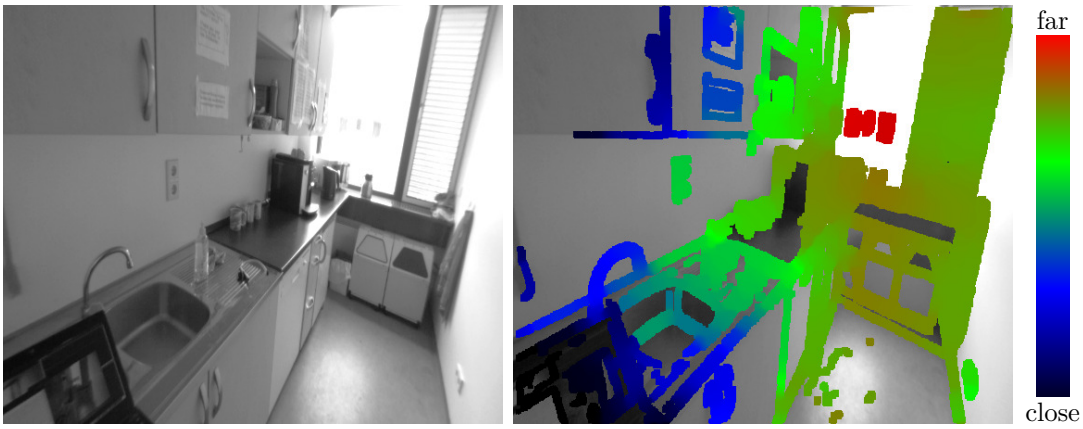


Figure 3.1: **Semi-Dense Monocular Visual Odometry.** Our approach works on a semi-dense inverse depth map and combines the accuracy and robustness of dense visual SLAM methods with the efficiency of feature-based techniques. Left: video frame, Right: color-coded semi-dense depth map, which consists of depth estimates in all image regions with sufficient structure.

tion step greatly reduces the complexity of the overall problem and allows it to be tackled in real time, it inherently comes with two significant drawbacks: First, only image information conforming to the respective feature type and parametrization – typically image corners and blobs [54] or line segments [68] – is utilized. Second, features have to be matched to each other, which often requires the costly computation of scale- and rotation-invariant descriptors and robust outlier estimation methods like RANSAC.

Dense monocular SLAM. To overcome these limitations and to better exploit the available image information, dense monocular SLAM methods [88, 122] have recently been proposed. The fundamental difference to keypoint-based approaches is that these methods directly work on the images instead of a set of extracted features, for both mapping and tracking: The world is modeled as dense surface while in turn new frames are tracked using whole-image alignment. This concept removes the need for discrete features, and allows to exploit all information present in the image, increasing tracking accuracy and robustness. To date however, doing this in real-time is only possible using modern, powerful GPU processors.

Similar methods are broadly used in combination with RGB-D cameras [66], which directly measure the depth of each pixel, or stereo camera rigs [26] – greatly reducing the complexity of the problem.

Dense multi-view stereo. Significant prior work exists on multi-view dense reconstruction, both in a real-time setting [88, 94, 112], as well as off-line [43, 100]. In particular for off-line reconstruction, there is a long history of using different

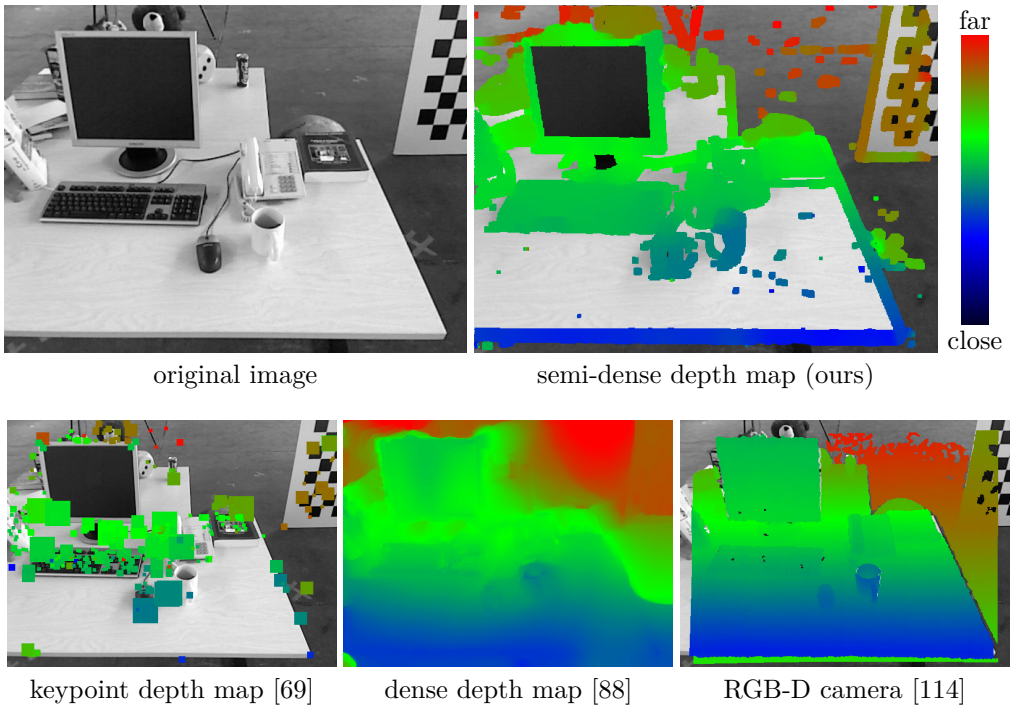


Figure 3.2: **Semi-Dense Approach.** Our approach reconstructs and tracks on a *semi-dense inverse depth map*, which is dense in all image regions carrying information (top-right). For comparison, the bottom row shows the respective result from a keypoint-based approach, a fully dense approach and the ground truth from an RGB-D camera.

baselines to steer the stereo-inherent trade-off between accuracy and precision [90]. Most similar to our approach is the early work of Matthies et al., who proposed probabilistic depth map fusion and propagation for image sequences [81], however only for structure from motion, i.e., not coupled with subsequent dense tracking.

3.1.2 Contributions

In this paper, we propose a novel semi-dense approach to monocular visual odometry, which does not require feature points. The key concepts are

- a probabilistic depth map representation,
- tracking based on whole-image alignment,
- the reduction on image-regions which carry information (semi-dense), and
- the full incorporation of stereo measurement uncertainty.

To the best of our knowledge, this is the first featureless, real-time monocular visual odometry approach, which runs in real-time on a CPU.

3.1.3 Method Outline

Our approach is partially motivated by the basic principle that for most real-time applications, video information is abundant and cheap to come by. Therefore, the computational budget should be spent such that the expected information gain is maximized. Instead of reducing the images to a sparse set of feature observations however, our method continuously estimates a *semi-dense inverse depth map* for the current frame, i.e., a dense depth map covering all image regions with non-negligible gradient (see Fig. 3.2). It is comprised of one inverse depth hypothesis per pixel modeled by a Gaussian probability distribution. This representation still allows to use whole-image alignment [66] to track new frames, while at the same time greatly reducing computational complexity compared to volumetric methods. The estimated depth map is propagated from frame to frame, and updated with variable-baseline stereo comparisons. We explicitly use prior knowledge about a pixel's depth to select a suitable reference frame on a per-pixel basis, and to limit the disparity search range.

The remainder of this paper is organized as follows: Section 3.2 describes the semi-dense mapping part of the proposed method, including the derivation of the observation accuracy as well as the probabilistic data fusion, propagation and regularization steps. Section 3.3 describes how new frames are tracked using whole-image alignment, and Sec. 3.4 summarizes the complete visual odometry method. A qualitative as well as a quantitative evaluation is presented in Sec. 3.5. We then give a brief conclusion in Sec. 3.6.

3.2 Semi-Dense Depth Map Estimation

One of the key ideas proposed in this paper is to estimate a semi-dense inverse depth map for the current camera image, which in turn can be used for estimating the camera pose of the next frame. This depth map is continuously propagated from frame to frame, and refined with new stereo depth measurements, which are obtained by performing per-pixel, adaptive-baseline stereo comparisons. This allows us to accurately estimate the depth both of close-by and far-away image regions. In contrast to previous work that accumulates the photometric cost over a sequence of several frames [88, 112], we keep exactly one inverse depth hypothesis per pixel that we represent as Gaussian probability distribution.

This section is comprised of three main parts: Section 3.2.1 describes the stereo method used to extract new depth measurements from previous frames, and how they are incorporated into the prior depth map. In Sec. 3.2.2, we describe how the depth map is propagated from frame to frame. In Sec. 3.2.3, we detail how we partially regularize the obtained depth map in each iteration, and how outliers are handled. Throughout this section, d denotes the *inverse* depth of a pixel.

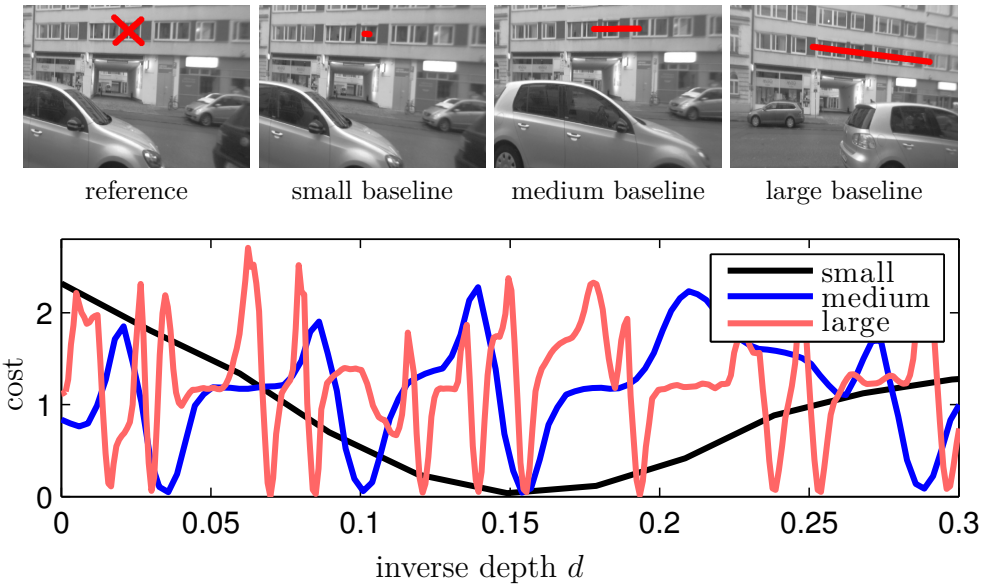


Figure 3.3: **Variable Baseline Stereo.** Reference image (left), three stereo images at different baselines (right), and the respective matching cost functions. While a small baseline (black) gives a unique, but imprecise minimum, a large baseline (red) allows for a very precise estimate, but has many false minima.

3.2.1 Stereo-Based Depth Map Update

It is well known [90] that for stereo, there is a trade-off between precision and accuracy (see Fig. 3.3). While many multiple-baseline stereo approaches resolve this by accumulating the respective cost functions over many frames [43, 94], we propose a probabilistic approach which explicitly takes advantage of the fact that in a video, small-baseline frames are available before large-baseline frames.

The full depth map update (performed once for each new frame) consists of the following steps: First, a subset of pixels is selected for which the accuracy of a disparity search is sufficiently large. For this we use three intuitive and very efficiently computable criteria, which will be derived in Sec. 3.2.1. For each selected pixel, we then individually select a suitable reference frame, and perform a one-dimensional disparity search. Propagated prior knowledge is used to reduce the disparity search range when possible, decreasing computational cost and eliminating false minima. The obtained inverse depth estimate is then fused into the depth map.

Reference Frame Selection

Ideally, the reference frame is chosen such that it maximizes the stereo accuracy, while keeping the disparity search range as well as the observation angle sufficiently small. As the stereo accuracy depends on many factors and because this selection is done for each pixel independently, we employ the following heuristic: We use the

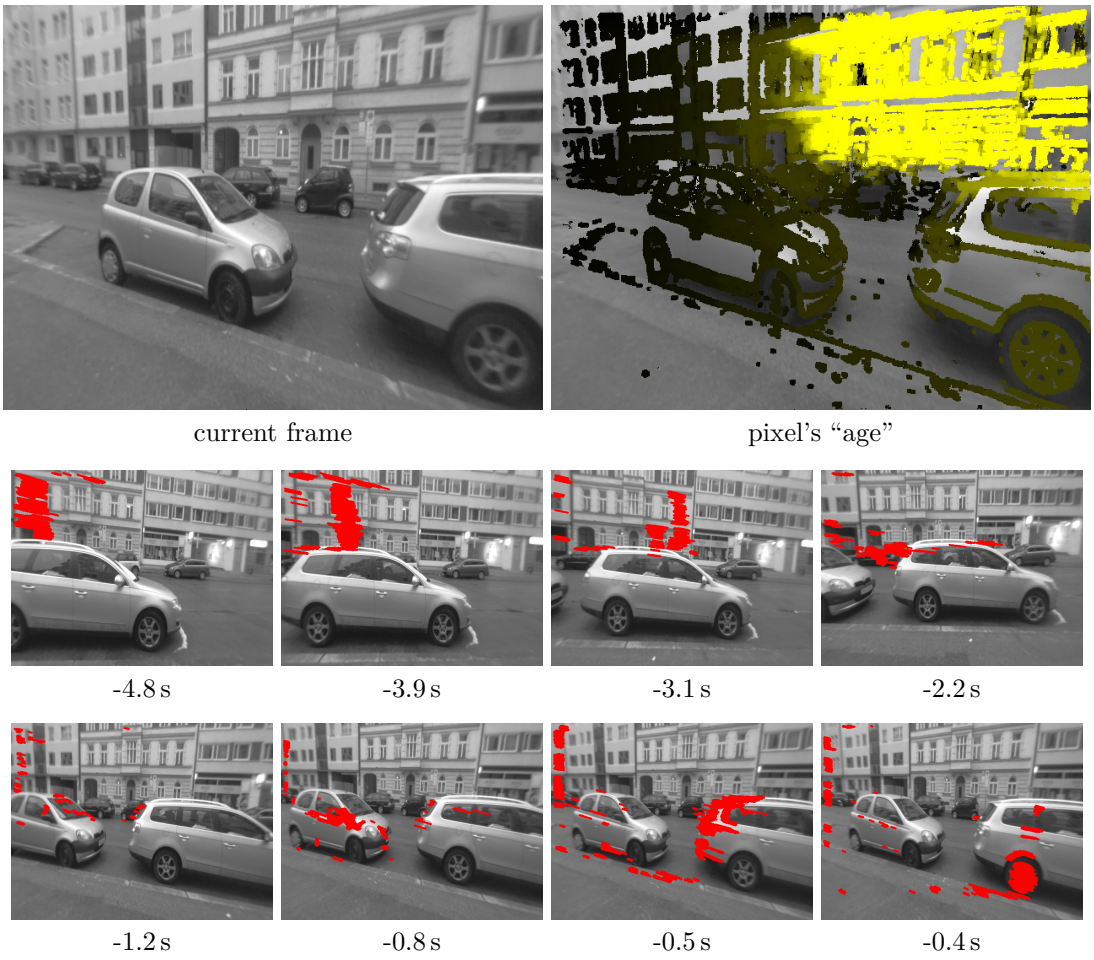


Figure 3.4: **Adaptive Baseline Selection.** For each pixel in the new frame (top left), a different stereo-reference frame is selected, based on how long the pixel was visible (top right: the more yellow, the older the pixel.). Some of the reference frames are displayed below, the red regions were used for stereo comparisons.

oldest frame the pixel was observed in, where the disparity search range and the observation angle do not exceed a certain threshold (see Fig. 3.4). If a disparity search is unsuccessful (i.e., no good match is found), the pixel's "age" is increased, such that subsequent disparity searches use newer frames where the pixel is likely to be still visible.

Stereo Matching Method

We perform an exhaustive search for the pixel's intensity along the epipolar line in the selected reference frame, and then perform a sub-pixel accurate localization of the matching disparity. If a prior inverse depth hypothesis is available, the search

interval is limited by $d \pm 2\sigma_d$, where d and σ_d denote the mean and standard deviation of the prior hypothesis. Otherwise, the full disparity range is searched.

In our implementation, we use the SSD error over five equidistant points on the epipolar line: While this significantly increases robustness in high-frequent image regions, it does not change the purely one-dimensional nature of this search. Furthermore, it is computationally efficient, as 4 out of 5 interpolated image values can be re-used for each SSD evaluation.

Uncertainty Estimation

In this section, we use uncertainty propagation to derive an expression for the error variance σ_d^2 on the inverse depth d. In general this can be done by expressing the optimal inverse depth d^* as a function of the noisy inputs – here we consider the images I_0, I_1 themselves, their relative orientation ξ and the camera calibration in terms of a projection function π[1]

$$d^* = d(I_0, I_1, \xi, \pi). \tag{3.1}$$

The error-variance of d^* is then given by

$$\sigma_d^2 = J_d \Sigma J_d^T, \tag{3.2}$$

where J_d is the Jacobian of d, and Σ the covariance of the input-error. For more details on covariance propagation, including the derivation of this formula, we refer to [25]. For simplicity, the following analysis is performed for patch-free stereo, i.e., we consider only a point-wise search for a *single intensity value* along the epipolar line.

For this analysis, we split the computation into three steps: First, the epipolar line in the reference frame is computed. Second, the best matching position $\lambda^* \in \mathbb{R}$ along it (i.e., the disparity) is determined. Third, the inverse depth d^* is computed from the disparity λ^*. The first two steps involve two independent error sources: the geometric error, which originates from noise on ξ and π and affects the first step, and the photometric error, which originates from noise in the images I_0, I_1 and affects the second step. The third step scales these errors by a factor, which depends on the baseline.

Geometric disparity error. The geometric error is the error ϵ_λ on the disparity λ^* caused by noise on ξ and π. While it would be possible to model, propagate, and estimate the complete covariance on ξ and π, we found that the gain in accuracy does not justify the increase in computational complexity. We therefore use an intuitive

[1] In the linear case, this is the camera matrix K – in practice however, nonlinear distortion and other (unmodeled) effects also play a role.

approximation: Let the considered epipolar line segment $L \subset \mathbb{R}^2$ be defined by

$$L := \left\{ l_0 + \lambda \begin{pmatrix} l_x \\ l_y \end{pmatrix} \mid \lambda \in S \right\}, \tag{3.3}$$

where λ is the disparity with search interval S, $(l_x, l_y)^T$ the *normalized* epipolar line direction and l_0 the point corresponding to infinite depth. We now assume that only the absolute position of this line segment, i.e., l_0 is subject to isotropic Gaussian noise ϵ_l. As in practice we keep the searched epipolar line segments short, the influence of rotational error is small, making this a good approximation.

Intuitively, a positioning error ϵ_l on the epipolar line causes a small disparity error ϵ_λ if the epipolar line is parallel to the image gradient, and a large one otherwise (see Fig. 3.5). This can be mathematically derived as follows: The image constrains the optimal disparity λ^* to lie on a certain isocurve, i.e. a curve of equal intensity. We approximate this isocurve to be locally linear, i.e., the gradient direction to be locally constant. This gives

$$l_0 + \lambda^* \begin{pmatrix} l_x \\ l_y \end{pmatrix} \stackrel{!}{=} g_0 + \gamma \begin{pmatrix} -g_y \\ g_x \end{pmatrix}, \qquad \gamma \in \mathbb{R} \tag{3.4}$$

where $g := (g_x, g_y)$ is the image gradient and g_0 a point on the isoline. The influence of noise on the image values will be derived in the next paragraph, hence at this point g and g_0 are assumed noise-free. Solving for λ gives the optimal disparity λ^* in terms of the noisy input l_0:

$$\lambda^*(l_0) = \frac{\langle g, g_0 - l_0 \rangle}{\langle g, l \rangle} \tag{3.5}$$

Analogously to (3.2), the variance of the geometric disparity error can then be expressed as

$$\sigma^2_{\lambda(\xi,\pi)} = J_{\lambda^*(l_0)} \begin{pmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_l^2 \end{pmatrix} J^T_{\lambda^*(l_0)} = \frac{\sigma_l^2}{\langle g, l \rangle^2}, \tag{3.6}$$

where g is the *normalized* image gradient, l the *normalized* epipolar line direction and σ_l^2 the variance of ϵ_l. Note that this error term solely originates from noise on the relative camera orientation ξ and the camera calibration π, i.e., it is independent of image intensity noise.

Photometric disparity error. Intuitively, this error encodes that small image intensity errors have a large effect on the estimated disparity if the image gradient is small, and a small effect otherwise (see Fig. 3.6). Mathematically, this relation can be derived as follows. We seek the disparity λ^* that minimizes the difference in intensities, i.e.,

$$\lambda^* = \min_\lambda \left(i_{\text{ref}} - I_p(\lambda) \right)^2, \tag{3.7}$$

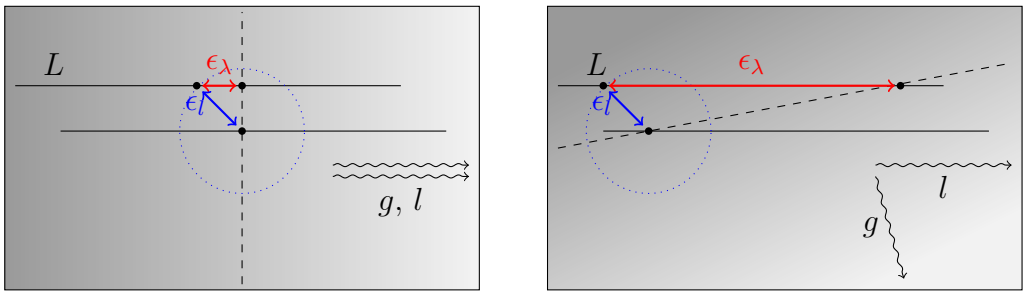


Figure 3.5: **Geometric Disparity Error.** Influence of a small positioning error ϵ_l of the epipolar line on the disparity error ϵ_λ. The dashed line represents the isocurve on which the matching point has to lie. ϵ_λ is small if the epipolar line is parallel to the image gradient (left), and a large otherwise (right).

where i_{ref} is the reference intensity, and $I_p(\lambda)$ the image intensity on the epipolar line at disparity λ. We assume a good initialization λ_0 to be available from the exhaustive search. Using a first-order Taylor approximation for I_p gives

$$\lambda^*(I) = \lambda_0 + (i_{\text{ref}} - I_p(\lambda_0))\, g_p^{-1}, \tag{3.8}$$

where g_p is the gradient of I_p, that is image gradient along the epipolar line. For clarity we only consider noise on i_{ref} and $I_p(\lambda_0)$; equivalent results are obtained in the general case when taking into account noise on the image values involved in the computation of g_p. The variance of the photometric disparity error is given by

$$\sigma^2_{\lambda(I)} = J_{\lambda^*(I)} \begin{pmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{pmatrix} J_{\lambda^*(I)} = \frac{2\sigma_i^2}{g_p^2}, \tag{3.9}$$

where σ_i^2 is the variance of the image intensity noise. The respective error originates solely from noisy image intensity values, and hence is independent of the geometric disparity error.

Pixel to inverse depth conversion. Using that, for small camera rotation, the inverse depth d is approximately proportional to the disparity λ, the observation variance of the inverse depth $\sigma^2_{d,\text{obs}}$ can be calculated using

$$\sigma^2_{d,\text{obs}} = \alpha^2 \left(\sigma^2_{\lambda(\xi,\pi)} + \sigma^2_{\lambda(I)} \right), \tag{3.10}$$

where the proportionality constant α – in the general, non-rectified case – is different for each pixel, and can be calculated from

$$\alpha := \frac{\delta_d}{\delta_\lambda}, \tag{3.11}$$

where δ_d is the length of the searched inverse depth interval, and δ_λ the length of the searched epipolar line segment. While α is inversely linear in the length of

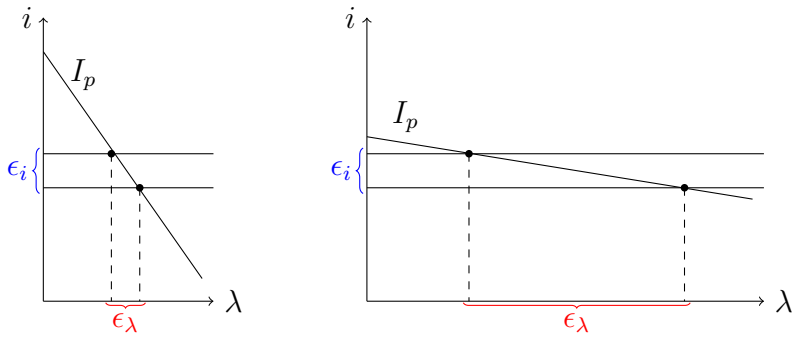


Figure 3.6: **Photometric Disparity Error.** Noise ϵ_i on the image intensity values causes a small disparity error ϵ_λ if the image gradient along the epipolar line is large (left). If the gradient is small, the disparity error is magnified (right).

the camera translation, it also depends on the translation direction and the pixel's location in the image.

When using an SSD error over multiple points along the epipolar line – as our implementation does – a good upper bound for the matching uncertainty is then given by

$$\sigma^2_{d,\text{obs-SSD}} \leq \alpha^2 \left(\min\{\sigma^2_{\lambda(\xi,\pi)}\} + \min\{\sigma^2_{\lambda(I)}\} \right), \tag{3.12}$$

where the min goes over all points included in the SSD error.

Depth Observation Fusion

After a depth observation for a pixel in the current image has been obtained, we integrate it into the depth map as follows: If no prior hypothesis for a pixel exists, we initialize it directly with the observation. Otherwise, the new observation is incorporated into the prior, i.e., the two distributions are multiplied (corresponding to the update step in a Kalman filter): Given a prior distribution $\mathcal{N}(d_p, \sigma_p^2)$ and a noisy observation $\mathcal{N}(d_o, \sigma_o^2)$, the posterior is given by

$$\mathcal{N}\left(\frac{\sigma_p^2 d_o + \sigma_o^2 d_p}{\sigma_p^2 + \sigma_o^2}, \frac{\sigma_p^2 \sigma_o^2}{\sigma_p^2 + \sigma_o^2} \right). \tag{3.13}$$

Summary of Uncertainty-Aware Stereo

New stereo observations are obtained on a per-pixel basis, adaptively selecting for each pixel a suitable reference frame and performing a one-dimensional search along the epipolar line. We identified the three major factors which determine the accuracy of such a stereo observation, i.e.,

- the **photometric disparity error** $\sigma^2_{\lambda(\xi,\pi)}$, depending on the *magnitude* of the image gradient along the epipolar line,

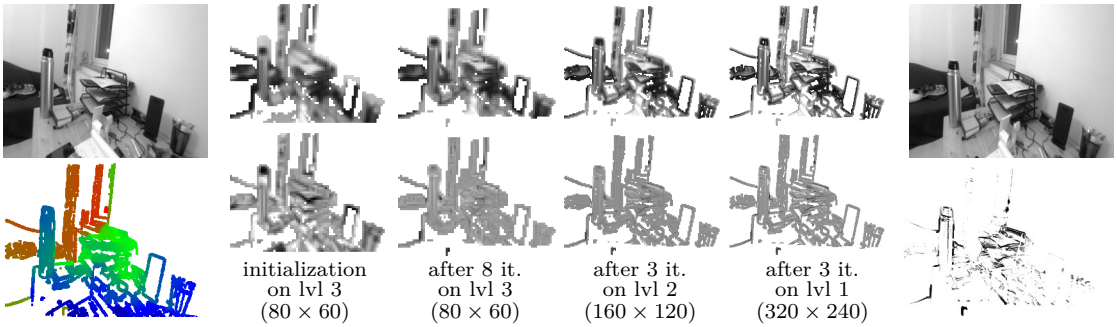


Figure 3.7: **Dense Tracking.** Reference image $I_1(\mathbf{x})$ (top left) with associated semi-dense inverse depth map (bottom left). The image in the top right shows the new frame $I_2(\mathbf{x})$ without depth information. Middle: Intermediate steps while minimizing $E(\xi)$ on different pyramid levels. The top row shows the back-warped new frame $I_2(w(\mathbf{x}, d, \xi))$, the bottom row shows the respective residual image $I_2(w(\mathbf{x}, d_i, \xi)) - I_1(\mathbf{x})$. The bottom right image shows the final pixel-weights (black = small weight). Small weights mainly correspond to newly occluded or disoccluded pixel.

- the **geometric disparity error** $\sigma^2_{\lambda(I)}$, depending on the *angle* between the image gradient and the epipolar line (independent of the gradient magnitude), and
- the **pixel to inverse depth ratio** α, depending on the camera translation, the focal length and the pixel's position.

These three simple-to-compute and purely local criteria are used to determine for which pixel a stereo update is worth the computational cost. Further, the computed observation variance is then used to integrate the new measurements into the existing depth map.

3.2.2 Depth Map Propagation

We continuously propagate the estimated inverse depth map from frame to frame, once the camera position of the next frame has been estimated. Based on the inverse depth estimate d_0 for a pixel, the corresponding 3D point is calculated and projected into the new frame, providing an inverse depth estimate d_1 in the new frame. The hypothesis is then assigned to the closest integer pixel position – to eliminate discretization errors, the sub-pixel accurate image location of the projected point is kept, and re-used for the next propagation step.

For propagating the inverse depth variance, we assume the camera rotation to be small. The new inverse depth d_1 can then be approximated by

$$d_1(d_0) = (d_0^{-1} - t_z)^{-1}, \tag{3.14}$$

where t_z is the camera translation along the optical axis. The variance of d_1 is hence given by

$$\sigma_{d_1}^2 = J_{d_1}\sigma_{d_0}^2 J_{d_1}^T + \sigma_p^2 = \left(\frac{d_1}{d_0}\right)^4 \sigma_{d_0}^2 + \sigma_p^2, \tag{3.15}$$

where σ_p^2 is the prediction uncertainty, which directly corresponds to the prediction step in an extended Kalman filter. It can also be interpreted as keeping the variance on the z-coordinate of a point fixed, i.e., setting $\sigma_{z_0}^2 = \sigma_{z_1}^2$. We found that using small values for σ_p^2 decreases drift, as it causes the estimated geometry to gradually "lock" into place.

Collision handling. At all times, we allow at most one inverse depth hypothesis per pixel: If two inverse depth hypothesis are propagated to the same pixel in the new frame, we distinguish between two cases:

1. if they are statistically similar, i.e., lie within 2σ bounds, they are treated as two independent observations of the pixel's depth and fused according to (3.13).

2. otherwise, the point that is further away from the camera is assumed to be occluded, and is removed.

3.2.3 Depth Map Regularization

For each frame – after all observations have been incorporated – we perform one regularization iteration by assigning each inverse depth value the average of the surrounding inverse depths, weighted by their respective inverse variance. To preserve sharp edges, if two adjacent inverse depth values are statistically different, i.e., are further away than 2σ, they do not contribute to one another. Note that the respective variances are not changed during regularization to account for the high correlation between neighboring hypotheses. Instead we use the minimal variance of all neighboring pixel when defining the stereo search range, and as a weighting factor for tracking (see Sec. 3.3).

Outlier removal. To handle outliers, we continuously keep track of the *validity* of each inverse depth hypothesis in terms of the probability that it is an outlier, or has become invalid (e.g., due to occlusion or a moving object). For each successful stereo observation, this probability is decreased. It is increased for each failed stereo search, if the respective intensity changes significantly on propagation, or when the absolute image gradient falls below a given threshold.

If, during regularization, the probability that all contributing neighbors are outliers – i.e., the product of their individual outlier-probabilities – rises above a given

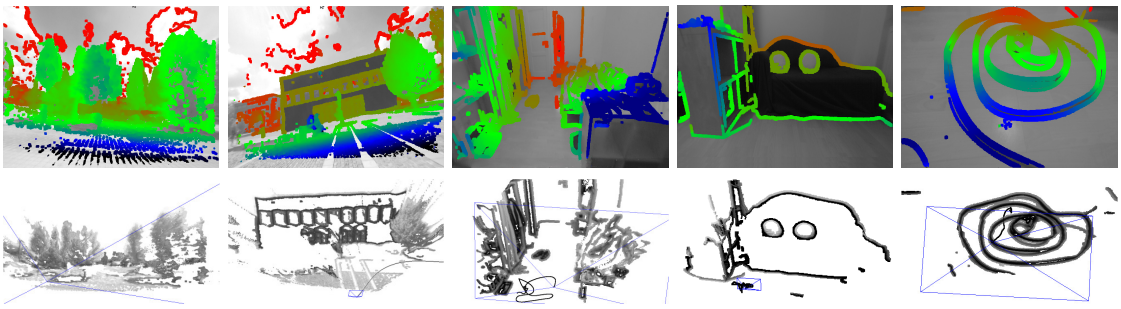


Figure 3.8: **Examples.** Top: Camera images overlaid with the respective estimated semi-dense inverse depth map. Bottom: 3D view of tracked scene. Note the versatility of our approach: It accurately reconstructs and tracks through (outside) scenes with a large depth-variance, including far-away objects like clouds , as well as (indoor) scenes with little structure and close to no image corners / keypoints. More examples are shown in the attached video.

threshold, the hypothesis is removed. Equally, if for an "empty" pixel this product drops below a given threshold, a new hypothesis is created from the neighbors. This fills holes arising from the forward-warping nature of the propagation step, and dilates the semi-dense depth map to a small neighborhood around sharp image intensity edges, which significantly increases tracking and mapping robustness.

3.3 Dense Tracking

Based on the inverse depth map of the previous frame, we estimate the camera pose of the current frame using dense image alignment. Such methods have previously been applied successfully (in real-time on a CPU) for tracking RGB-D cameras [66], which directly provide dense depth measurements along with the color image. It is based on the direct minimization of the photometric error

$$r_i(\xi) := \left(I_2(w(\mathbf{x}_i, d_i, \xi)) - I_1(\mathbf{x}_i)\right)^2, \tag{3.16}$$

where the warp function $w\colon \Omega_1 \times \mathbb{R} \times \mathbb{R}^6 \to \Omega_2$ maps each point $\mathbf{x}_i \in \Omega_1$ in the reference image I_1 to the respective point $w(\mathbf{x}_i, d_i, \xi) \in \Omega_2$ in the new image I_2. As input it requires the 3D pose of the camera $\xi \in \mathbb{R}^6$ and uses the estimated inverse depth $d_i \in \mathbb{R}$ for the pixel in I_1. Note that no depth information with respect to I_2 is required.

To increase robustness to self-occlusion and moving objects, we apply a weighting scheme as proposed in [66]. Further, we add the variance of the inverse depth $\sigma^2_{d_i}$ as an additional weighting term, making the tracking resistant to recently initialized and still inaccurate depth estimates from the mapping process. The final energy

that is minimized is hence given by

$$E(\xi) := \sum_i \frac{\alpha(r_i(\xi))}{\sigma_{d_i}^2} r_i(\xi), \tag{3.17}$$

where $\alpha \colon \mathbb{R} \to \mathbb{R}$ defines the weight for a given residual. Minimizing this error can be interpreted as computing the maximum likelihood estimator for ξ, assuming independent noise on the image intensity values. The resulting weighted least-squares problem is solved efficiently using an iteratively reweighted Gauss-Newton algorithm coupled with a coarse-to-fine approach, using four pyramid levels. Figure 3.7 shows an example of the tracking process. For further details on the minimization we refer to [16].

3.4 System Overview

Tracking and depth estimation is split into two separate threads: One continuously propagates the inverse depth map to the most recent tracked frame, updates it with stereo-comparisons and partially regularizes it. The other simultaneously tracks each incoming frame on the most recent available depth map. While tracking is performed in real-time at 30Hz, one complete mapping iteration takes longer and is hence done at roughly 15Hz – if the map is heavily populated, we adaptively reduce the number of stereo comparisons to maintain a constant frame-rate. For stereo observations, a buffer of up to 100 past frames is kept, automatically removing those that are used least.

We use a standard, keypoint-based method to obtain the relative camera pose between two initial frames, which are then used to initialize the inverse depth map needed for tracking successive frames. From this point onward, our method is entirely self-contained. In preliminary experiments, we found that in most cases our approach is even able to recover from random or extremely inaccurate initial depth maps, indicating that the keypoint-based initialization might become superfluous in the future.

3.5 Results

We have tested our approach on both publicly available benchmark sequences, as well as live, using a hand-held camera. Some examples are shown in Fig. 3.8. Note that our method does not attempt to build a global map, i.e., once a point leaves the field of view of the camera or becomes occluded, the respective depth value is deleted. All experiments are performed on a standard consumer laptop with Intel i7 quad-core CPU. In a preprocessing step, we rectify all images such that a pinhole camera-model can be applied.

Table 3.1: **Results on RGB-D Benchmark.**

	position drift (cm/s)			**rotation drift** (deg/s)		
	ours	[66]	[69]	ours	[66]	[69]
fr2/xyz	0.6	0.6	8.2	0.33	0.34	3.27
fr2/desk	2.1	2.0	-	0.65	0.70	-

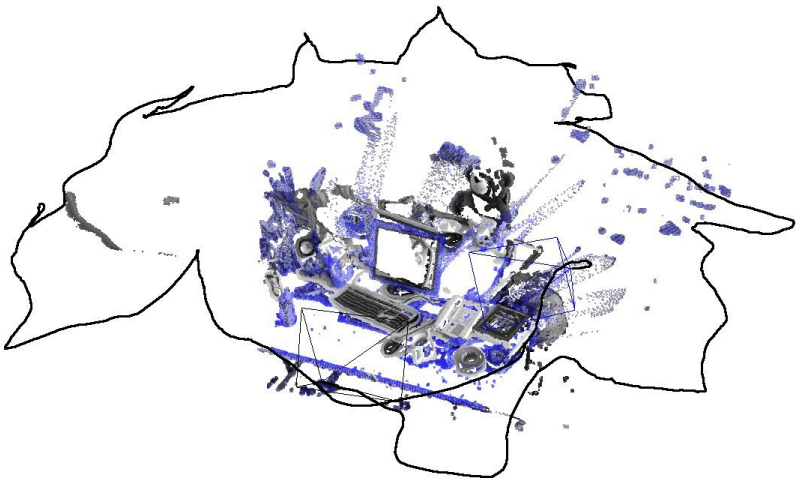


Figure 3.9: **RGB-D Benchmark Sequence fr2/desk.** Tracked camera trajectory (black), the depth map of the first frame (blue), and the estimated depth map (gray-scale) after a complete loop around the table. Note how well certain details such as the keyboard and the monitor align.

3.5.1 RGB-D Benchmark Sequences

As basis for a quantitative evaluation and to facilitate reproducibility and easy comparison with other methods, we use the TUM RGB-D benchmark [114]. For tracking and mapping we only use the gray-scale images; for the very first frame however the provided depth image is used as initialization.

Our method (like any monocular visual odometry method) fails in case of pure camera rotation, as the depth of new regions cannot be determined. The achieved tracking accuracy for two feasible sequences – that is, sequences which do not contain strong camera rotation without simultaneous translation – is given in Table 3.1. For comparison we also list the accuracy from (1) a state-of-the-art, dense RGB-D odometry [66], and (2) a state-of-the-art, keypoint-based monocular SLAM system (PTAM, [69]). We initialize PTAM using the built-in stereo initializer, and perform a 7DoF (rigid body plus scale) alignment to the ground truth trajectory. Figure 3.9 shows the tracked camera trajectory for fr2/desk. We found that our method achieves similar accuracy as [66] which uses the same dense tracking algorithm but relies on the Kinect depth images. The keypoint-based approach [69] proves to be

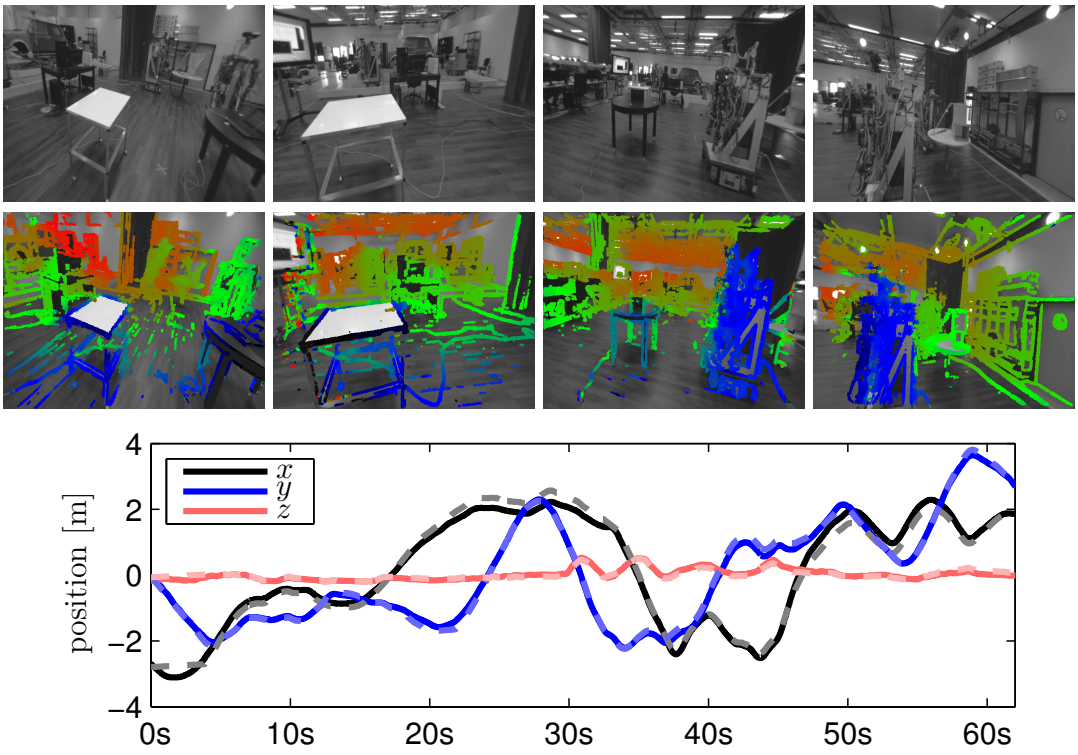


Figure 3.10: **Additional Sequence.** Estimated camera trajectory and ground truth (dashed) for a long and challenging sequence. The complete sequence is shown in the attached video.

significantly less accurate and robust; it consistently failed after a few seconds for the second sequence.

3.5.2 Additional Test Sequences

To analyze our approach in more detail, we recorded additional challenging sequences with the corresponding ground truth trajectory in a motion capture studio. Figure 3.10 shows an extract from the video, as well as the tracked and the ground-truth camera position over time. As can be seen from the figure, our approach is able to maintain a reasonably dense depth map at all times and the estimated camera trajectory matches closely the ground truth.

3.6 Conclusion

In this paper we proposed a novel visual odometry method for a monocular camera, which does not require discrete features. In contrast to previous work on dense tracking and mapping, our approach is based on probabilistic depth map estimation and fusion over time. Depth measurements are obtained from patch-free stereo matching in different reference frames at a suitable baseline, which are selected on a per-pixel basis. To our knowledge, this is the first featureless monocular visual odometry method which runs in real-time on a CPU. In our experiments, we showed that the tracking performance of our approach is comparable to that of fully dense methods without requiring a depth sensor.

Chapter 4

LSD-SLAM: Large-Scale Direct Monocular SLAM

Authors	Jakob Engel[1]	engelj@in.tum.de
	Thomas Schöps[2]	schoepst@inf.ethz.ch
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] ETH Zürich	
Publication	**LSD-SLAM: Large-Scale Direct Monocular SLAM**. J. Engel, T. Schöps, D. Cremers. In *Preceedings of European Conference on Computer Vision (ECCV), 2014*. Copyright 2013 Springer. Reprinted with permission from Springer. doi: 10.1007/978-3-319-10605-2_54	
Contribution	Problem definition	*significantly contributed*
	Literature survey	*significantly contributed*
	Method development & evaluation	*significantly contributed*
	Implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

Abstract We propose a direct (feature-less) monocular SLAM algorithm which, in contrast to current state-of-the-art regarding direct methods, allows to build large-scale, consistent maps of the environment. Along with highly accurate pose estimation based on direct image alignment, the 3D environment is reconstructed in real-time as pose-graph of keyframes with associated semi-dense depth maps. These are obtained by filtering over a large number of pixelwise small-baseline stereo comparisons. The explicitly scale-drift aware formulation allows the approach to operate on challenging sequences including large variations in scene scale. Major enablers are two key novelties: (1) a novel direct tracking method which operates on $\mathfrak{sim}(3)$, thereby explicitly detecting scale-drift, and (2) an elegant probabilistic solution to include the effect of noisy depth values into tracking. The resulting direct monocular SLAM system runs in real-time on a CPU.

4.1 Introduction

Real-time monocular Simultaneous Localization and Mapping (SLAM) and 3D reconstruction have become increasingly popular research topics. Two major reasons are (1) their use in robotics, in particular to navigate unmanned aerial vehicles (UAVs) [14, 7, 41], and (2) augmented and virtual reality applications slowly making their way into the mass-market.

One of the major benefits of *monocular* SLAM – and simultaneously one of the biggest challenges – comes with the inherent scale-ambiguity: The scale of the world cannot be observed and drifts over time, being one of the major error sources. The advantage is that this allows to seamlessly switch between differently scaled environments, such as a desk environment indoors and large-scale outdoor environments. Scaled sensors on the other hand, such as depth or stereo cameras, have a limited range at which they can provide reliable measurements and hence do not provide this flexibility.

4.1.1 Related Work

Feature-Based Methods.

The fundamental idea behind feature-based approaches (both filtering-based [69, 78] and keyframe-based [69]) is to split the overall problem – estimating geometric information from images – into two sequential steps: First, a set of feature observations is extracted from the image. Second, the camera position and scene geometry is computed as a function of these feature observations only.

While this decoupling simplifies the overall problem, it comes with an important limitation: *Only information that conforms to the feature type can be used.* In

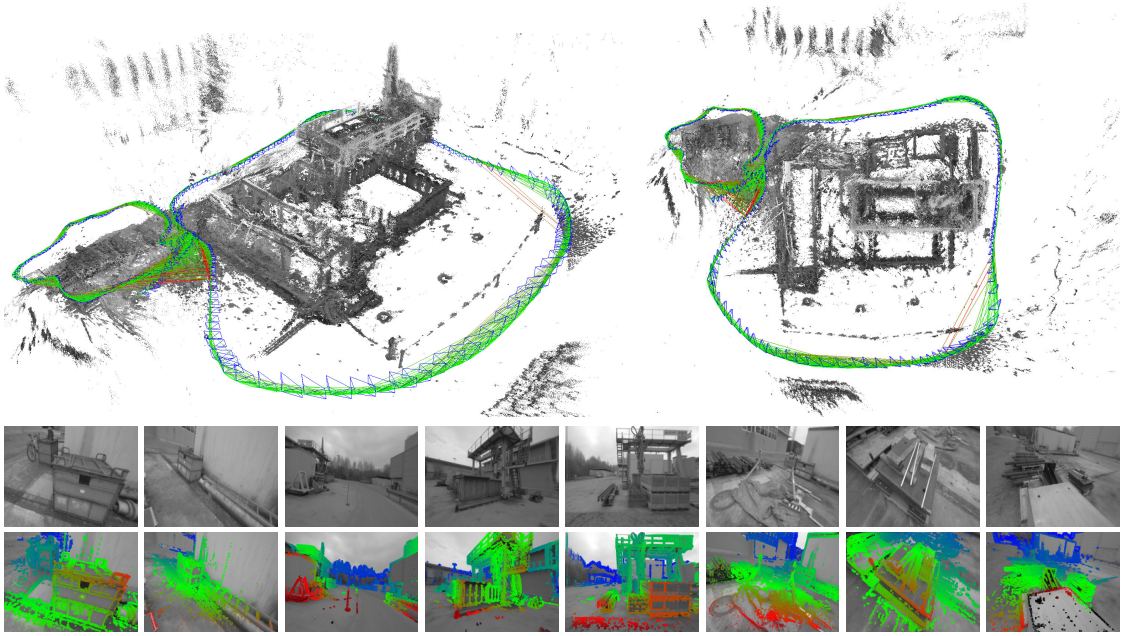


Figure 4.1: **Large-Scale Direct Monocular SLAM.** LSD-SLAM generates a consistent global map, using direct image alignment and probabilistic, semi-dense depth maps instead of keypoints. Top: Accumulated pointclouds of all keyframes of a medium-sized trajectory (from a hand-held monocular camera), generated in real-time. Bottom: A selection of keyframes with color-coded semi-dense inverse depth map. See also the supplementary video.

particular, when using keypoints, information contained in straight or curved edges – which especially in man-made environments make up a large part of the image – is discarded. Several approaches have been made in the past to remedy this by including edge-based [36, 68] or even region-based [27] features. Yet, since the estimation of the high-dimensional feature space is tedious, they are rarely used in practice. To obtain dense reconstructions, the estimated camera poses can be used to subsequently reconstruct dense maps, using multiview stereo [15].

Direct Methods.

Direct visual odometry (VO) methods circumvent this limitation by optimizing the geometry directly on the image intensities, which *enables using all information in the image*. In addition to higher accuracy and robustness in particular in environments with little keypoints, this provides substantially more information about the geometry of the environment, which can be very valuable for robotics or augmented reality applications.

While direct image alignment is well-established for RGB-D or stereo sensors [26, 65], only recently monocular direct VO algorithms have been proposed: In [88, 93, 112], accurate and fully dense depth maps are computed using a variational

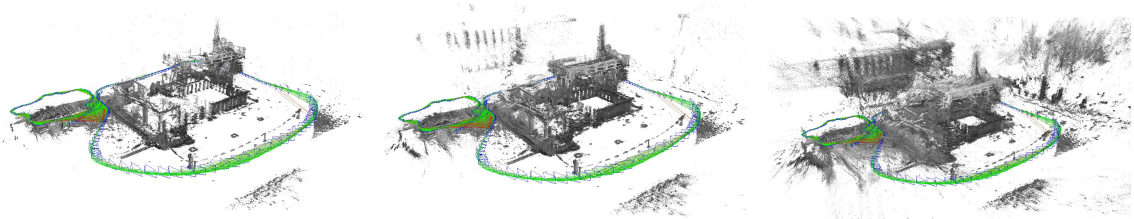


Figure 4.2: **Variance Estimation.** In addition to accurate, semi-dense 3D reconstructions, LSD-SLAM also estimates the associated uncertainty. From left to right: Accumulated pointcloud thesholded with different maximum variance. Note how the reconstruction becomes significantly more dense, but at the same time includes more noise.

formulation, which however is computationally demanding and requires a state-of-the-art GPU to run in real-time. In [9], a semi-dense depth filtering formulation was proposed which significantly reduces computational complexity, allowing real-time operation on a CPU and even on a modern smartphone [11]. By combining direct tracking with keypoints, [41] achieves high frame-rates even on embedded platforms. All these approaches however are pure visual odometries, they only locally track the motion of the camera and do not build a consistent, global map of the environment including loop-closures.

Pose Graph Optimization.

This is a well-known SLAM technique to build a consistent, global map: The world is represented as a number of keyframes connected by pose-pose constraints, which can be optimized using a generic graph optimization framework like g2o [73].

In [65], a pose graph based RGB-D SLAM method is proposed, which also incorporates geometric error to allow tracking through scenes with little texture. To account for scale-drift arising in monocular SLAM, [109] proposed a keypoint-based monocular SLAM system which represents camera poses as 3D similarity transforms instead of rigid body movements.

4.1.2 Contributions and Outline

We propose a **L**arge-**S**cale **D**irect monocular **SLAM** (LSD-SLAM) method, which not only locally tracks the motion of the camera, but allows to build consistent, large-scale maps of the environment (see Fig. 4.1 and 4.2). The method uses direct image alignment coupled with filtering-based estimation of semi-dense depth maps as originally proposed in [9]. The global map is represented as a pose graph consisting of keyframes as vertices with 3D similarity transforms as edges, elegantly incorporating changing scale of the environment and allowing to detect and correct accumulated drift. The method runs in real-time on a CPU, and as odometry even on a modern smartphone [11]. The main contributions of this

paper are (1) a framework for large-scale, direct monocular SLAM, in particular a novel scale-aware image alignment algorithm to directly estimate the similarity transform $\boldsymbol{\xi} \in \mathfrak{sim}(3)$ between two keyframes, and (2) probabilistically consistent incorporation of uncertainty of the estimated depth into tracking.

4.2 Preliminaries

In this chapter we give a condensed summary of the relevant mathematical concepts and notation. In particular, we summarize the representation of 3D poses as elements of Lie-Algebras (Sec. 4.2.1), derive direct image alignment as weighted least-squares minimization on Lie-manifolds (Sec. 4.2.2), and briefly introduce propagation of uncertainty (Sec. 4.2.3).

Notation.

We denote matrices by bold, capital letters ($\mathbf{R}$) and vectors as bold, lower case letters ($\boldsymbol{\xi}$). The n'th row of a matrix is denoted by $[\cdot]_n$. Images $I\colon \Omega \to \mathbb{R}$, the per-pixel inverse depth map $D\colon \Omega \to \mathbb{R}^+$ and the inverse depth variance map $V\colon \Omega \to \mathbb{R}^+$ are written as functions, where $\Omega \subset \mathbb{R}^2$ is the set of *normalized* pixel coordinates, i.e., they include the intrinsic camera calibration. Throughout the paper we use d to denote the *inverse* of the depth z of a point, i.e., $d = z^{-1}$.

4.2.1 3D Rigid Body and Similarity Transformations

3D Rigid Body Transformations.

A 3D rigid body transform $\mathbf{G} \in \mathrm{SE}(3)$ denotes rotation and translation in 3D, i.e., is defined by

$$\mathbf{G} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \quad \text{with} \quad \mathbf{R} \in \mathrm{SO}(3) \text{ and } \mathbf{t} \in \mathbb{R}^3. \tag{4.1}$$

During optimization, a minimal representation for the camera pose is required, which is given by the corresponding element $\boldsymbol{\xi} \in \mathfrak{se}(3)$ of the associated Lie-algebra. Elements are mapped to SE(3) by the exponential map $\mathbf{G} = \exp_{\mathfrak{se}(3)}(\boldsymbol{\xi})$, its inverse being denoted by $\boldsymbol{\xi} = \log_{\mathrm{SE}(3)}(\mathbf{G})$. With a slight abuse of notation, we consistently use elements of $\mathfrak{se}(3)$ to represent poses, which we directly write as vector $\boldsymbol{\xi} \in \mathbb{R}^6$. The transformation moving a point from frame i to frame j is written as $\boldsymbol{\xi}_{ji}$. For convenience, we define the pose concatenation operator $\circ\colon \mathfrak{se}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$ as

$$\boldsymbol{\xi}_{ki} := \boldsymbol{\xi}_{kj} \circ \boldsymbol{\xi}_{ji} := \log_{\mathrm{SE}(3)} \left(\exp_{\mathfrak{se}(3)}(\boldsymbol{\xi}_{kj}) \cdot \exp_{\mathfrak{se}(3)}(\boldsymbol{\xi}_{ji}) \right). \tag{4.2}$$

Further, we define the 3D projective warp function ω, which projects an image point $\mathbf{p}$ and its inverse depth d into a by $\boldsymbol{\xi}$ transformed camera frame

$$\omega(\mathbf{p}, d, \boldsymbol{\xi}) := \begin{pmatrix} x'/z' \\ y'/z' \\ 1/z' \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} := \exp_{\mathfrak{se}(3)}(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{p}_x/d \\ \mathbf{p}_y/d \\ 1/d \\ 1 \end{pmatrix}. \tag{4.3}$$

3D Similarity Transformations.

A 3D similarity transform $\mathbf{S} \in \mathrm{Sim}(3)$ denotes rotation, scaling and translation, i.e., is defined by

$$\mathbf{S} = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \quad \text{with} \quad \mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3 \ \text{ and } \ s \in \mathbb{R}^+. \tag{4.4}$$

As for rigid body transformations, a minimal representation is given by elements of the associated Lie-algebra $\boldsymbol{\xi} \in \mathfrak{sim}(3)$, which now have an additional degree of freedom, that is $\boldsymbol{\xi} \in \mathbb{R}^7$. The exponential and logarithmic map, pose concatenation and a projective warp function ω_s can be defined analogously to the $\mathfrak{se}(3)$ case, for further details see [109].

4.2.2 Weighted Gauss-Newton Optimization on Lie-Manifolds

Two images are aligned by Gauss-Newton minimization of the photometric error

$$E(\boldsymbol{\xi}) = \sum_i \underbrace{\left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \boldsymbol{\xi}))\right)^2}_{=:r_i^2(\boldsymbol{\xi})}, \tag{4.5}$$

which gives the maximum-likelihood estimator for $\boldsymbol{\xi}$ assuming i.i.d. Gaussian residuals. We use a left-compositional formulation: Starting with an initial estimate $\boldsymbol{\xi}^{(0)}$, in each iteration a left-multiplied increment $\delta\boldsymbol{\xi}^{(n)}$ is computed by solving for the minimum of a Gauss-Newton second-order approximation of E:

$$\delta\boldsymbol{\xi}^{(n)} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}(\boldsymbol{\xi}^{(n)}) \ \text{ with } \ \mathbf{J} = \left.\frac{\partial \mathbf{r}(\boldsymbol{\epsilon} \circ \boldsymbol{\xi}^{(n)})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}, \tag{4.6}$$

where $\mathbf{J}$ is the derivative of the stacked residual vector $\mathbf{r} = (r_1, \ldots, r_n)^T$ with respect to a left-multiplied increment, and $\mathbf{J}^T\mathbf{J}$ the Gauss-Newton approximation of the Hessian of E. The new estimate is then obtained by multiplication with the computed update

$$\boldsymbol{\xi}^{(n+1)} = \delta\boldsymbol{\xi}^{(n)} \circ \boldsymbol{\xi}^{(n)}. \tag{4.7}$$

In order to be robust to outliers arising e.g. from occlusions or reflections, different weighting-schemes [65] have been proposed, resulting in an iteratively reweighted least-squares problem: In each iteration, a weight matrix $\mathbf{W} = \mathbf{W}(\boldsymbol{\xi}^{(n)})$ is computed which down-weights large residuals. The iteratively solved error function then becomes

$$E(\boldsymbol{\xi}) = \sum_i w_i(\boldsymbol{\xi}) r_i^2(\boldsymbol{\xi}), \tag{4.8}$$

and the update is computed as

$$\delta\boldsymbol{\xi}^{(n)} = -(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}r(\boldsymbol{\xi}^{(n)}). \tag{4.9}$$

Assuming the residuals to be independent, the inverse of the Hessian from the last iteration $(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}$ is an estimate for the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ of a left-multiplied error onto the final result, that is

$$\boldsymbol{\xi}^{(n)} = \boldsymbol{\epsilon} \circ \boldsymbol{\xi}_{\text{true}} \quad \text{with} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}}). \tag{4.10}$$

In practice, the residuals are highly correlated, such that $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ is only a lower bound - yet it contains valuable information about the correlation between noise on the different degrees of freedom. Note that we follow a **left-multiplication** convention, equivalent results can be obtained using a right-multiplication convention. However, the estimated covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ depends on the multiplication order – when used in a pose graph optimization framework, this has to be taken into account. The left-multiplication convention used here is consistent with [109], while e.g. the default type-implementation in g2o [73] assumes right-multiplication.

4.2.3 Propagation of Uncertainty

Propagation of uncertainty is a statistical tool to derive the uncertainty of the output of a function $f(\mathbf{X})$, caused by uncertainty on its input $\mathbf{X}$. Assuming $\mathbf{X}$ to be Gaussian distributed with covariance $\boldsymbol{\Sigma}_{\mathbf{X}}$, the covariance of $f(\mathbf{X})$ can be approximated (using the Jacobian $\mathbf{J}_f$ of f) by

$$\boldsymbol{\Sigma}_f \approx \mathbf{J}_f \boldsymbol{\Sigma}_{\mathbf{X}} \mathbf{J}_f^T. \tag{4.11}$$

4.3 Large-Scale Direct Monocular SLAM

We start by giving an overview of the complete algorithm in Sec. 4.3.1, and briefly introduce the representation for the global map in Sec. 4.3.2. The three main components of the algorithm are then described in Sec. 4.3.3 (tracking of new frames), Sec. 4.3.4 (depth map estimation), Sec. 4.3.5 (keyframe-to-keyframe tracking) and finally Sec. 4.3.6 (map optimization).

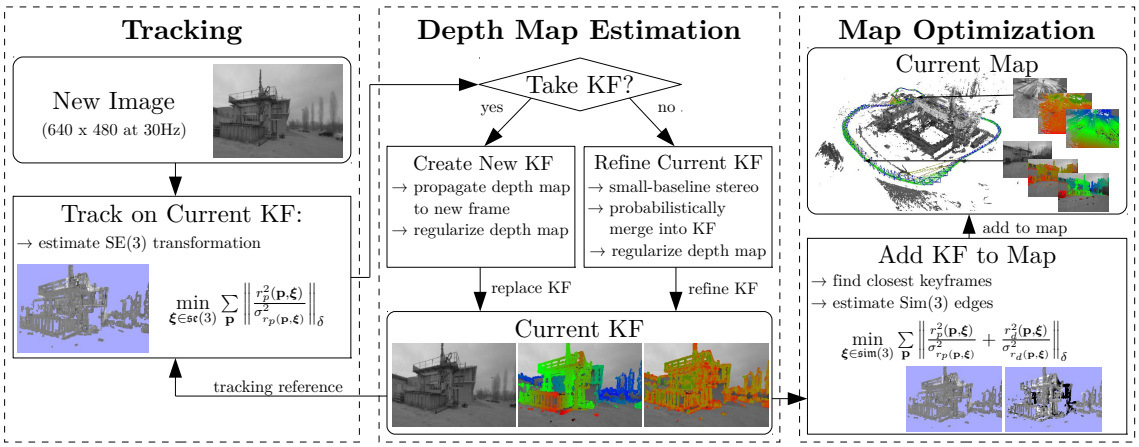


Figure 4.3: **Algorithm Overview.** Overview over the complete LSD-SLAM algorithm.

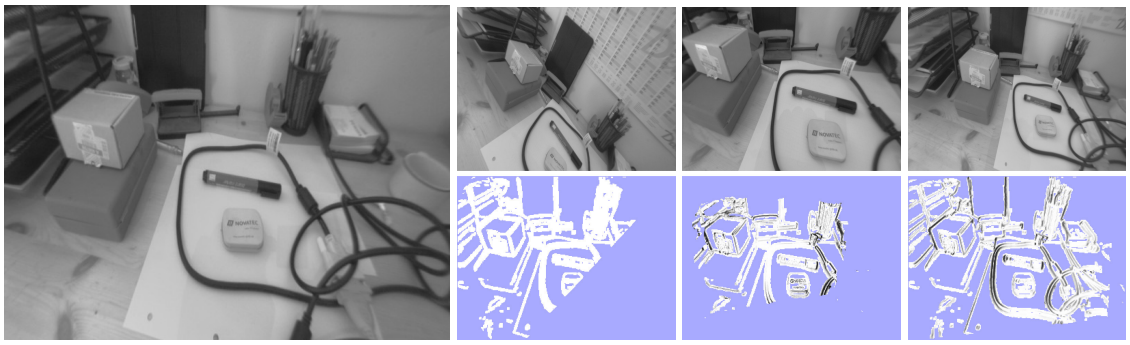
4.3.1 The Complete Method

The algorithm consists of three major components: **tracking**, **depth map estimation** and **map optimization** as visualized in Fig. 4.3:

- The **tracking** component continuously tracks new camera images. That is, it estimates their rigid body pose $\boldsymbol{\xi} \in \mathfrak{se}(3)$ with respect to the current keyframe, using the pose of the previous frame as initialization.
- The **depth map estimation** component uses tracked frames to either refine or replace the current keyframe. Depth is refined by filtering over many per-pixel, small-baseline stereo comparisons coupled with interleaved spatial regularization as originally proposed in [9]. If the camera has moved too far, a new keyframe is initialized by projecting points from existing, close-by keyframes into it.
- Once a keyframe is replaced as tracking reference – and hence its depth map will not be refined further – it is incorporated into the global map by the **map optimization** component. To detect loop closures and scale-drift, a similarity transform $\boldsymbol{\xi} \in \mathfrak{sim}(3)$ to close-by existing keyframes (including its direct predecessor) is estimated using scale-aware, direct $\mathfrak{sim}(3)$-image alignment.

Initialization.

To bootstrap the LSD-SLAM system, it is sufficient to initialize a first keyframe with a *random* depth map and large variance. Given sufficient translational camera movement in the first seconds, the algorithm "locks" to a certain configuration, and after a couple of keyframe propagations converges to a correct depth configuration. Some examples are shown in the attached video. A more thorough evaluation of



(a) reference image (b) rotation (c) z trans. (d) x trans.

Figure 4.4: **Statistic normalization.** (a) reference image. (b-d): tracked images and inverse variance $\sigma_{r_p}^{-2}$ of the residual. For pure rotation, depth noise has no effect on the residual noise and hence all normalization factors are the same. For z translation depth noise has no effect for pixels in the center of the image, while for x translation it only affects residuals with intensity-gradient in x direction.

this ability to converge without dedicated initial bootstrapping is outside the scope of this paper, and remains for future work.

4.3.2 Map Representation

The map is represented as a pose graph of keyframes: Each keyframe $\mathcal{K}_i$ consists of a camera image $I_i \colon \Omega_i \to \mathbb{R}$, an inverse depth map $D_i \colon \Omega_{Di} \to \mathbb{R}^+$, and the variance of the inverse depth $V_i \colon \Omega_{Di} \to \mathbb{R}^+$. Note that the depth map and variance are only defined for a subset of pixels $\Omega_{Di} \subset \Omega_i$, containing all image regions in the vicinity of sufficiently large intensity gradient, hence **semi-dense**. Edges $\mathcal{E}_{ji}$ between keyframes contain their relative alignment as similarity transform $\boldsymbol{\xi}_{ji} \in \mathfrak{sim}(3)$, as well as the corresponding covariance matrix $\boldsymbol{\Sigma}_{ji}$.

4.3.3 Tracking new Frames: Direct $\mathfrak{se}(3)$ Image Alignment

Starting from an existing keyframe $\mathcal{K}_i = (I_i, D_i, V_i)$, the relative 3D pose $\boldsymbol{\xi}_{ji} \in \mathfrak{se}(3)$ of a new image I_j is computed by minimizing the variance-normalized photometric error

$$E_p(\boldsymbol{\xi}_{ji}) = \sum_{\mathbf{p} \in \Omega_{D_i}} \left\| \frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}} \right\|_\delta \tag{4.12}$$

$$\text{with} \quad r_p(\mathbf{p}, \boldsymbol{\xi}_{ji}) := I_i(\mathbf{p}) - I_j(\omega(\mathbf{p}, D_i(\mathbf{p}), \boldsymbol{\xi}_{ji})) \tag{4.13}$$

$$\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})} := 2\sigma_I^2 + \left(\frac{\partial r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\partial D_i(\mathbf{p})} \right)^2 V_i(\mathbf{p}) \tag{4.14}$$

where $\| \cdot \|_\delta$ is the Huber norm

$$\|r^2\|_\delta := \begin{cases} \frac{r^2}{2\delta} & \text{if } |r| \leq \delta \\ |r| - \frac{\delta}{2} & \text{otherwise.} \end{cases} \tag{4.15}$$

applied to the *normalized* residual. The residual's variance $\sigma^2_{r_p(\mathbf{p},\boldsymbol{\xi}_{ji})}$ is computed using covariance propagation as described in Sec. 4.2.3, and utilizing the inverse depth variance V_i. Further, we assume Gaussian image intensity noise σ_I^2. Minimization is performed using iteratively re-weighted Gauss-Newton optimization as described in Sec. 4.2.2.

In contrast to previous direct methods, the proposed formulation explicitly takes into account varying noise on the depth estimates: This is of particular relevance as for direct, monocular SLAM, this noise differs significantly for different pixels, depending on how long they were visible – which is in contrast to approaches working on RGB-D data, for which the uncertainty on the inverse depth is approximately constant. Figure 4.4 shows how this weighting behaves for different types of motion. Note that no depth information for the new camera image is available – therefore, the scale of the new image is not defined, and the minimization is performed on $\mathfrak{se}(3)$.

4.3.4 Depth Map Estimation

Keyframe Selection.

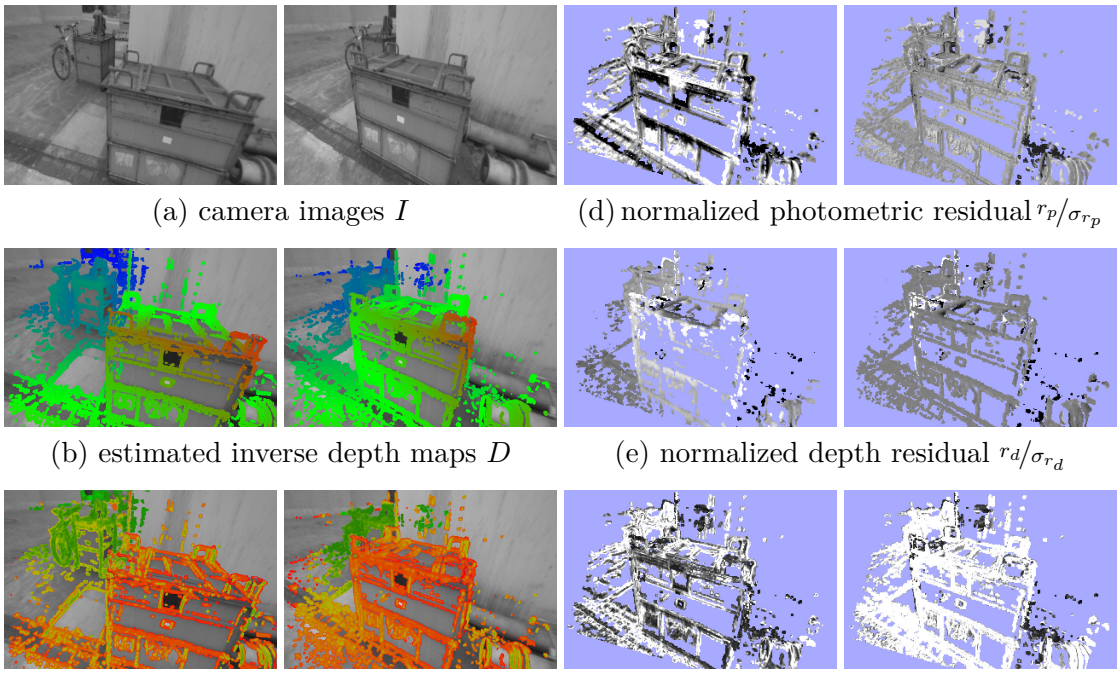
If the camera moves too far away from the existing map, a new keyframe is created from the most recent tracked image. We threshold a weighted combination of relative distance and angle to the current keyframe:

$$\text{dist}(\boldsymbol{\xi}_{ji}) := \boldsymbol{\xi}_{ji}^T \mathbf{W} \boldsymbol{\xi}_{ji} \tag{4.16}$$

where $\mathbf{W}$ is a diagonal matrix containing the weights. Note that, as described in the following section, each keyframe is scaled such that its mean inverse depth is one. This threshold is therefore relative to the current scale of the scene, and ensures sufficient possibilities for small-baseline stereo comparisons.

Depth Map Creation.

Once a new frame is chosen to become a keyframe, its depth map is initialized by projecting points from the previous keyframe into it, followed by one iteration of spatial regularization and outlier removal as proposed in [9]. Afterwards, the depth map is scaled to have a mean inverse depth of one - this scaling factor is directly incorporated into the $\mathfrak{sim}(3)$ camera pose. Finally, it replaces the previous keyframe and is used for tracking subsequent new frames.



(a) camera images I (d) normalized photometric residual r_p/σ_{r_p}

(b) estimated inverse depth maps D (e) normalized depth residual r_d/σ_{r_d}

(c) inverse depth variance V (f) robust Huber weights

Figure 4.5: **Direct keyframe alignment on $\mathfrak{sim}(3)$.** (a)-(c): two keyframes with associated depth and depth variance. (d)-(f): photometric residual, depth residual and Huber weights, before minimization (left), and after minimization (right).

Depth Map Refinement.

Tracked frames that do not become a keyframe are used to refine the current keyframe: A high number of very efficient small-baseline stereo comparisons is performed for image regions where the expected stereo accuracy is sufficiently large, as described in [9]. The result is incorporated into the existing depth map, thereby refining it and potentially adding new pixels – this is done using the filtering approach proposed in [9].

4.3.5 Constraint Acquisition: Direct $\mathfrak{sim}(3)$ Image Alignment

Direct Image Alignment on $\mathfrak{sim}(3)$.

Monocular SLAM is – in contrast to RGB-D or Stereo-SLAM – inherently scale-ambivalent, i.e., the absolute scale of the world is not observable. Over long trajectories this leads to scale-drift, which is one of the major sources of error [109]. Further, all distances are only defined up to scale, which causes threshold-based outlier rejection or parametrized robust kernels (e.g. Huber) to be ill-defined. We solve

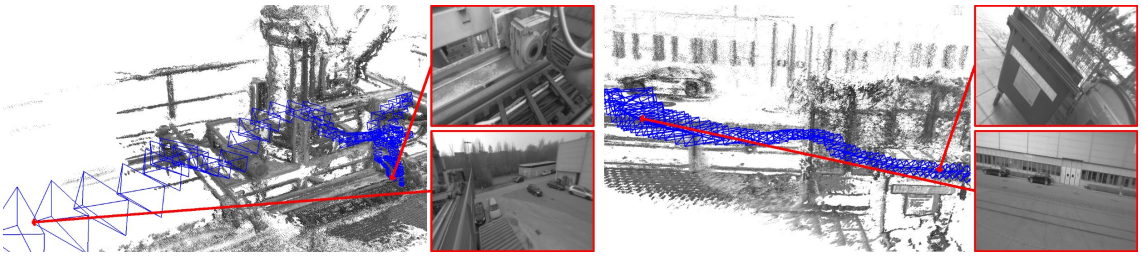


Figure 4.6: **Two scenes with high scale variation.** Camera frustums are displayed for each keyframe with their size corresponding to the keyframe's scale.

this by using the inherent correlation between scene depth and tracking accuracy: The depth map of each created keyframe is scaled such that the mean inverse depth is one. In return, edges between keyframes are estimated as elements of $\mathfrak{sim}(3)$, elegantly incorporating the scaling difference between keyframes, and, in particular for large loop-closures, allowing an explicit detection of accumulated scale-drift.

For this, we propose a novel method to perform *direct, scale-drift aware image alignment on* $\mathfrak{sim}(3)$, which is used to align two differently scaled keyframes. In addition to the photometric residual r_p, we incorporate a depth residual r_d which penalizes deviations in inverse depth between keyframes, allowing to directly estimate the scaled transformation between them. The total error function that is minimized becomes

$$E(\boldsymbol{\xi}_{ji}) := \sum_{\mathbf{p}\in\Omega_{D_i}} \left\| \frac{r_p^2(\mathbf{p},\boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p},\boldsymbol{\xi}_{ji})}} + \frac{r_d^2(\mathbf{p},\boldsymbol{\xi}_{ji})}{\sigma^2_{r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}} \right\|_\delta, \tag{4.17}$$

where the photometric residual r_p^2 and $\sigma^2_{r_p}$ is defined as in (4.13) - (4.14). The depth residual and its variance is computed as

$$r_d(\mathbf{p},\boldsymbol{\xi}_{ji}) := [\mathbf{p}']_3 - D_j([\mathbf{p}']_{1,2}) \tag{4.18}$$

$$\sigma^2_{r_d(\mathbf{p},\boldsymbol{\xi}_{ji})} := V_j([\mathbf{p}']_{1,2}) \left(\frac{\partial r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}{\partial D_j([\mathbf{p}']_{1,2})} \right)^2 + V_i(\mathbf{p}) \left(\frac{\partial r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}{\partial D_i(\mathbf{p})} \right)^2, \tag{4.19}$$

where $\mathbf{p}' := \omega_s(\mathbf{p}, D_i(\mathbf{p}), \boldsymbol{\xi}_{ji})$ denotes the transformed point. Note that the Huber norm is applied to the sum of the normalized photometric and depth residual – which accounts for the fact that if one is an outlier, the other typically is as well. Note that for tracking on $\mathfrak{sim}(3)$, the inclusion of the depth error is **required** as the photometric error alone does not constrain the scale. Minimization is performed analogously to direct image alignment on $\mathfrak{se}(3)$ using the iteratively re-weighted Gauss-Newton algorithm (Sec. 4.2.2). In practice, $\mathfrak{sim}(3)$ tracking is computationally only marginally more expensive than tracking on $\mathfrak{se}(3)$, as only little additional computations are needed[1].

[1]We approximate the gradient of the depth map to be zero, which significantly speeds up the computation

Constraint Search.

After a new keyframe $\mathcal{K}_i$ is added to the map, a number of possible loop closure keyframes $\mathcal{K}_{j_1}, ..., \mathcal{K}_{j_n}$ is collected: We use the closest ten keyframes, as well as a suitable candidate proposed by an appearance-based mapping algorithm [47] to detect large-scale loop closures. To avoid insertion of false or falsely tracked loop closures, we then perform a **reciprocal tracking check**: For each candidate $\mathcal{K}_{j_k}$ we independently track $\boldsymbol{\xi}_{j_k i}$ and $\boldsymbol{\xi}_{i j_k}$. Only if the two estimates are statistically similar, i.e., if

$$e(\boldsymbol{\xi}_{j_k i}, \boldsymbol{\xi}_{i j_k}) := (\boldsymbol{\xi}_{j_k i} \circ \boldsymbol{\xi}_{i j_k})^T \left(\boldsymbol{\Sigma}_{j_k i} + \mathrm{Adj}_{j_k i} \boldsymbol{\Sigma}_{i j_k} \mathrm{Adj}_{j_k i}^T \right)^{-1} (\boldsymbol{\xi}_{j_k i} \circ \boldsymbol{\xi}_{i j_k}) \tag{4.20}$$

is sufficiently small, they are added to the global map. For this, the adjoint $\mathrm{Adj}_{j_k i}$ is used to transform $\boldsymbol{\Sigma}_{i j_k}$ into the correct tangent space.

Convergence Radius for $\mathfrak{sim}(3)$ Tracking.

An important limitation of direct image alignment lies in the inherent non-convexity of the problem, and hence the need for a sufficiently accurate initialization. While for the tracking of new camera frames a sufficiently good initialization is available (given by the pose of the previous frame), this is not the case when finding loop closure constraints, in particular for large loop closures.

One solution for this consists in using a very small number of keypoints to compute a better initialization: Using the depth values from the existing inverse depth maps, this requires aligning two sets of 3D points with known correspondences, which can be done efficiently in closed form using e.g. the method of Horn [55]. Still, we found that in practice the convergence radius is sufficiently large even for large-scale loop closures - in particular we found that the convergence radius can be substantially increased by the following measures:

- **Efficient Second Order Minimization (ESM)** [19]: While our results confirm previous work [71] in that ESM *does not significantly* increase the precision of dense image alignment, we observed that it *does* slightly increase the convergence radius.
- **Coarse-to-Fine Approach**: While a pyramid approach is commonly used for direct image alignment, we found that starting at a very low resolution of only 20×15 pixels – much smaller than usually done – already helps to increase the convergence radius.

An evaluation of the effect of these measures is given in Sec. 4.4.3.

4.3.6 Map Optimization

The map, consisting of a set of keyframes and tracked $\mathfrak{sim}(3)$-constraints, is continuously optimized in the background using pose graph optimization [73]. The error

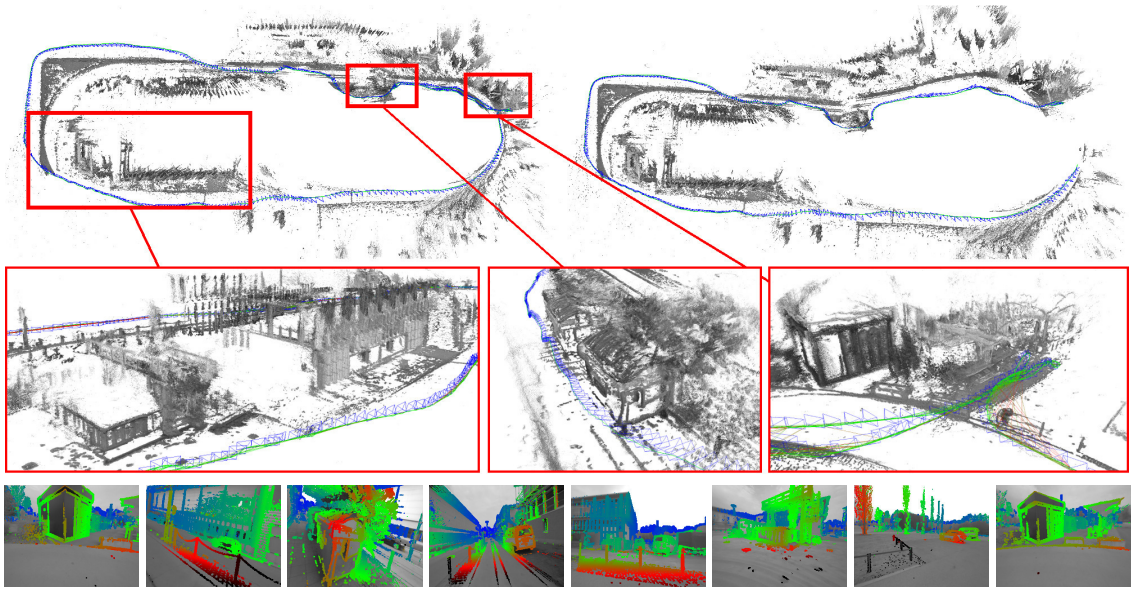


Figure 4.7: **Large Loop closure.** Loop closure for a long and challenging outdoor trajectory (after the loop closure on the left, before on the right). Also shown are three selected close-ups of the generated pointcloud, and semi-dense depth maps for selected keyframes.

function that is minimized is – in accordance with the left-multiplication convention from Sec. 4.2.2 – defined by (W defining the world frame)

$$E(\boldsymbol{\xi}_{W1} \dots \boldsymbol{\xi}_{Wn}) := \sum_{(\boldsymbol{\xi}_{ji}, \boldsymbol{\Sigma}_{ji}) \in \mathcal{E}} (\boldsymbol{\xi}_{ji} \circ \boldsymbol{\xi}_{Wi}^{-1} \circ \boldsymbol{\xi}_{Wj})^T \boldsymbol{\Sigma}_{ji}^{-1} (\boldsymbol{\xi}_{ji} \circ \boldsymbol{\xi}_{Wi}^{-1} \circ \boldsymbol{\xi}_{Wj}). \tag{4.21}$$

4.4 Results

We evaluate LSD-SLAM both quantitatively on publicly available datasets [51, 114] as well as on challenging outdoor trajectories, recorded with a hand-held monocular camera. Some of the evaluated trajectories are shown in full in the supplementary video.

4.4.1 Qualitative Results on Large Trajectories

We tested the algorithm on several long and challenging trajectories, which include many camera rotations, large scale changes and major loop closures. Figure 4.7 shows a roughly 500 m long trajectory which takes 6 minutes just before and after the large loop closure is found. Figure 4.8 shows a challenging trajectory with large variations in scene depth, which also includes a loop closure.

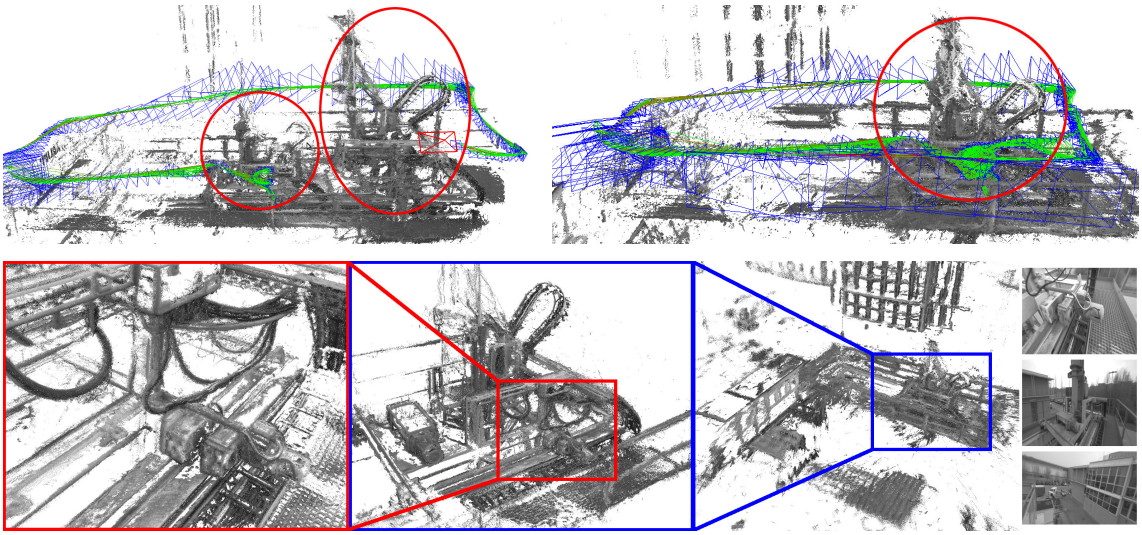


Figure 4.8: **Loop closure with explicit scale correction.** Accumulated pointcloud of a trajectory with large scale variation, including views with an average inverse depth of less than 20 cm to more than 10 m. After the loop closure (top-right), the geometry is consistently aligned, while before (top-left) parts of the scene existed twice, at different scales. The bottom row shows different close-ups of the scene. The proposed scale-aware formulation allows to accurately estimate both fine details and large-scale geometry – this flexibility is one of the major benefits of a monocular approach.

4.4.2 Quantitative Evaluation

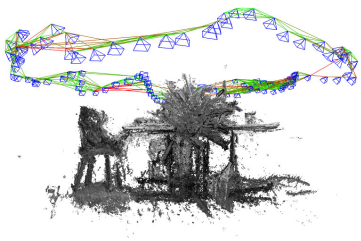
We evaluate LSD-SLAM on the publicly available RGB-D dataset [114]. Note that for monocular SLAM this is a very challenging benchmark, as it contains fast rotational movement, strong motion blur and rolling shutter artifacts. We use the very first depth map to bootstrap the system and get the correct initial scale. Table 4.9 shows the resulting absolute trajectory error, and compares it to other approaches.

4.4.3 Convergence Radius for $\mathfrak{sim}(3)$ Tracking

We evaluate the convergence radius on two exemplary sequences, the result is shown in Fig. 4.10. Even though direct image alignment is non-convex, we found that with the steps proposed in Sec. 4.3.5, surprisingly large camera movements can be tracked. It can also be observed that these measures only increase the convergence radius, and have no notable effect on tracking precision.

4.5 Conclusion

We have presented a novel direct (feature-less) monocular SLAM algorithm which we call LSD-SLAM, which runs in real-time on a CPU. In contrast to existing direct



	LSD-SLAM (#KF)	[9]	[69]	[65]	[37]
fr2/desk	4.52 (116)	13.50	x	1.77	9.5
fr2/xyz	1.47 (38)	3.79	24.28	1.18	2.6
sim/desk	0.04 (39)	1.53	-	0.27	-
sim/slowmo	0.35 (12)	2.21	-	0.13	-

Figure 4.9: **Quantitative Evaluation.** Results on the TUM RGB-D benchmark [114], and two simulated sequences from [51], measured as absolute trajectory RMSE (cm). For LSD-SLAM, we also show the number of keyframes created. 'x' denotes tracking failure, '-' no available data. For comparison we show respective results from semi-dense mono-VO [9], keypoint-based mono-SLAM [69], direct RGB-D SLAM [65] and keypoint-based RGB-D SLAM [37]. Note that [65] and [37] use depth information from the sensor, while the others do not.

approaches – which are all pure odometries – it maintains and tracks on a global map of the environment, which contains a pose-graph of keyframes with associated probabilistic semi-dense depth maps. Major components of the proposed method are two key novelties: (1) a direct method to align two keyframes on $\mathfrak{sim}(3)$, explicitly incorporating and detecting scale-drift and (2) a novel, probabilistic approach to incorporate noise on the estimated depth maps into tracking. Represented as point clouds, the map gives a semi-dense and highly accurate 3D reconstruction of the environment. We experimentally showed that the approach reliably tracks and maps even challenging hand-held trajectories with a length of over 500 m, in particular including large variations in scale within the same sequence (*average* inverse depth of less than 20 cm to more than 10 m) and large rotations – demonstrating its versatility, robustness and flexibility.

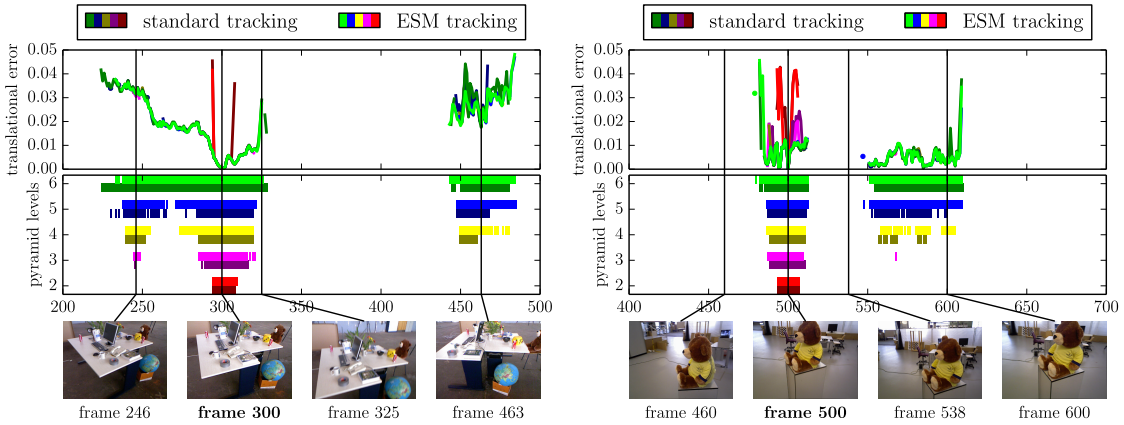


Figure 4.10: **Convergence Radius Analysis.** Convergence radius and accuracy of $\mathfrak{sim}(3)$ direct image alignment with and without ESM minimization (indicated by light / dark) for a different number of pyramid levels (color). All frames of the respective sequence are tracked on frame 300 (left) and frame 500 (right), using the identity as initialization. The bottom plots show for which frames tracking succeeds; the top plots show the final translational error. ESM and more pyramid levels clearly increase the convergence radius, however these measures have no notable effect on tracking precision: if tracking converges, it almost always converges to the same minimum.

Chapter 5

Semi-Dense Visual Odometry for AR on a Smartphone

Authors	Thomas Schöps[2]	schoepst@inf.ethz.ch
	Jakob Engel[1]	engelj@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] ETH Zürich	

Publication **Semi-Dense Visual Odometry for AR on a Smartphone**. T. Schöps, J. Engel, D. Cremers. In *Preceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2014*.

doi: 10.1109/ISMAR.2014.6948420

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

Abstract We present a direct monocular visual odometry system which runs in real-time on a smartphone. Being a direct method, it tracks and maps on the images themselves instead of extracted features such as keypoints. New images are tracked using direct image alignment, while geometry is represented in the form of a semi-dense depth map. Depth is estimated by filtering over many small-baseline, pixel-wise stereo comparisons. This leads to significantly less outliers and allows to map and use all image regions with sufficient gradient, including edges. We show how a simple world model for AR applications can be derived from semi-dense depth maps, and demonstrate the practical applicability in the context of an AR application in which simulated objects can collide with real geometry.

5.1 Introduction

Estimating the movement of a monocular camera and the 3D structure of the environment is amongst the most prominent challenges in computer vision. Commonly referred to as monocular SLAM or structure from motion, it is a key enabler for many augmented reality applications: only if the precise pose of the camera is available in real-time, virtual objects can be rendered into the scene as if they were part of it. Further, knowledge about the geometry of the scene allows virtual objects to interact with it: in an augmented reality game, game characters can collide with, be occluded by or be placed on top of real obstacles. To assist with furnishing or re-decorating a room, a piece of furniture could be reconstructed from a video taken by a smartphone, and virtually rendered into different locations in the room. Figure 5.1 shows an example AR application realized on top of our direct Visual Odometry (VO) system.

Apart from marker based methods [40, 120, 121] – which allow for precise and fast camera pose estimation at the cost of having to manually place one or more physical markers into the scene – state-of-the-art monocular SLAM methods generally operate on features. While this allows to estimate the camera movement in real-time on mobile platforms [70, 85], the resulting feature based maps hardly provide sufficient information about the 3D geometry of the scene for physical interaction.

At the same time, recent advances in computer vision have shown the high potential of *direct* methods for monocular SLAM [9, 41, 88, 93]: instead of operating on features, these methods perform both tracking and mapping directly on the image intensity values. Fundamentally different from feature based methods, direct methods not only allow for fast, sub-pixel accurate camera tracking, but also provide substantially more information about the 3D structure of the environment, are less susceptible to outliers, and more robust in environments with little texture [9].

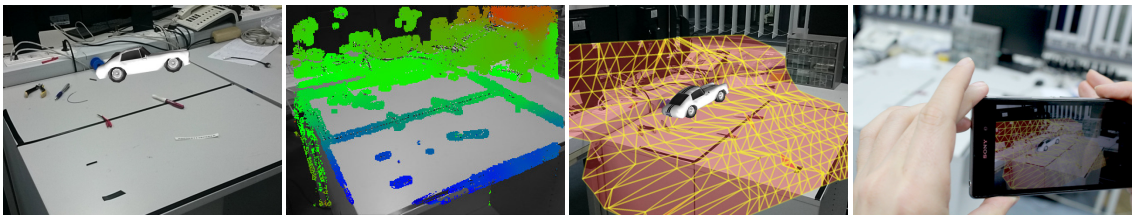


Figure 5.1: **Semi-Dense VO on a smartphone.** From left to right: AR demo application with simulated car. Corresponding estimated semi-dense depth map. Estimated dense collision mesh, fixed and shown from a different perspective. Photo of running system. The attached video shows the system in action.

5.1.1 Related Work

In this section we give an overview over existing monocular SLAM and VO methods, divided into *feature based* and *direct* methods. While there exists a large number of feature based methods for mobile phones, existing direct methods are computationally expensive and require a powerful GPU to run in real-time. Figure 5.2 summarizes the main differences between feature based and direct methods.

Feature Based. The basic idea behind features is to split the overall problem – estimating geometric information from images – into two separate, sequential steps: First, a set of feature observations is extracted from the image, typically independently of one another. This can be done using a large variety of methods, including different corner detectors and descriptors, as well as fast matching methods and outlier detection schemes like RANSAC. Second, camera position and scene geometry are computed as a function of these feature observations only. Again, there exists a variety of methods to do this, including bundle adjustment based approaches [69] or filtering based approaches [31, 78].

While decoupling image based (photometric) estimation from subsequent geometric estimation simplifies the overall problem, it comes with an important limitation: *Only information that conforms to the feature type and parametrization can be used.* In particular, when using keypoints, information contained in edges is discarded.

Today, there are several keypoint based monocular VO and SLAM methods which run in real-time on mobile devices [70, 78]. In order to obtain a denser 3D reconstruction, one approach is to perform two-frame or multi-frame stereo on selected frames, where the camera pose is obtained from a full feature based SLAM system running in the background [115]. However – even though the computed dense depth maps are often more accurate and precise than the feature based map, they cannot directly be fed back into the SLAM system running in the background, thereby discarding valuable information.

Direct. Direct approaches circumvent these limitations by directly optimizing

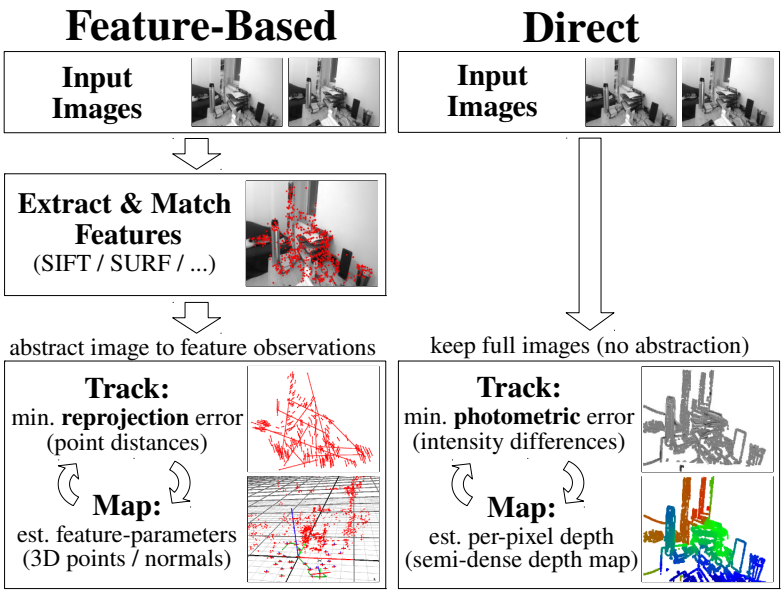


Figure 5.2: **Semi-Dense vs. Keypoints.** Feature based methods abstract images to feature observations and discard all other information. In contrast, the proposed direct approach maps and tracks directly on image intensities: this allows to (1) use all information, including e.g. edges and (2) directly obtain rich, semi-dense information about the geometry of the scene.

the camera poses and scene geometry on the raw images. *This allows to use all information in the image*, leading to higher accuracy and robustness in particular in indoor environments with only few features. Early direct or semi-direct approaches were based on scene representations by sets of planar patches: [106] presents such a system which simultaneously estimates the motion, scene structure and illumination. [61] also combines tracking and reconstruction and especially discusses local optimums in the error function.

Images are tracked by direct minimization of the per-pixel photometric error (see Sec. 5.2.1), which is well established for tracking RGB-D or stereo sensors [26, 65]. In a monocular setting, the required per-pixel depth values are in turn computed from stereo on previous frames: in [9], a pixel-wise filtering formulation was proposed, which fuses information from many small-baseline stereo comparisons. This approach allows to obtain accurate and precise semi-dense depth maps in real-time on a CPU. It has recently been extended to LSD-SLAM, a large-scale direct monocular SLAM system [4] including loop-closures. Another approach is to compute fully dense depth maps using a variational formulation [88, 93, 112], which however is computationally demanding and requires a state-of-the-art GPU to run in real-time. A common feature of direct methods is their inherent parallelism: as many operations are defined on a per-pixel basis, they can ideally be parallelized using GPGPUs or SIMD (Single Instruction Multiple Data) instructions, achieving

Figure 5.3: **Examples of semi-dense depth maps.** Estimated in real-time on a smartphone. See also the attached video.

considerable speed-ups in practice.

5.1.2 Contributions and Outline

In this paper we present a direct monocular VO system based on [9] which runs in real-time on a smartphone. In addition to accurately computing the camera pose at over 30 Hz, the proposed method provides rich information about the environment in the form of a semi-dense depth map of the currently visible scene. In particular we (1) describe modifications required to run the algorithm in real-time on a smartphone, and (2) propose a method to derive a dense world model suitable for basic physical interaction of simulated objects with the real world. We demonstrate the capabilities of the proposed approach with a simple augmented reality game, in which a simulated car drives through the environment, and can collide with real obstacles in the scene.

The paper is organized as follows: We describe the proposed semi-dense, direct VO method in Sec. 5.2. In particular, in Sec. 5.2.3, we describe the steps required for real-time performance on a smartphone. Following this, we show how collision meshes are computed from the semi-dense depth map in Sec. 5.3, and how they are used in a simple AR application. Finally, we qualitatively evaluate the resulting system in different environments in Sec. 5.4.

5.2 Semi-Dense Direct Visual Odometry

The proposed monocular VO algorithm does not use features at any stage of the algorithm, but instead directly operates on the raw intensity images: The map is represented as a *semi-dense inverse depth map*, which contains a Gaussian probability distribution (mean and variance) for the inverse depth of a subset of pixels, hence "semi-dense". An example is shown in Fig. 5.3: pixels that have a depth hypothesis are shown in color (encoding the depth).

The whole system is divided into two parts (see Fig. 5.4), running in parallel: **tracking** and **mapping**. In Sec. 5.2.1, we describe tracking using direct image alignment. In Sec. 5.2.2, we present the mapping part which simultaneously estimates and propagates the depth map. The system is closely based on the approach by Engel et al. [9] for real-time operation on a consumer laptop.

Notation. We represent an image as function $I\colon \Omega \to \mathbb{R}$. Similarly, we represent the inverse depth map and inverse depth variance map as functions $D\colon \Omega_D \to \mathbb{R}^+$ and $V\colon \Omega_D \to \mathbb{R}^+$, where Ω_D contains all pixels which have a valid depth hypothesis. Note that D and V denote mean and variance of the *inverse* depth, as this approximates the uncertainty of stereo much better than assuming a Gaussian-distributed depth.

Initialization. We initialize the map with random depth values and large variance for the first frame. When moving the camera slowly and in parallel to the image plane, the algorithm (running normally) typically locks onto a consistent depth configuration and quickly converges to a valid map. This is in contrast to [9], in which a keypoint based initializer was used. While the process is successful in most cases, we observed one distinct failure case which results in an inverse estimate of both the depth map and the camera motion, which is further discussed in [61]. A numerical evaluation of the initialization success and convergence rate is given in Sec. 5.4.

5.2.1 Tracking

The pose of new frames is estimated using direct image alignment: given the current map $\{I_M,\, D_M,\, V_M\}$, the relative pose $\boldsymbol{\xi} \in \mathrm{SE}(3)$ of a new frame I is obtained by directly minimizing the photometric error

$$E(\boldsymbol{\xi}) := \sum_{\mathbf{x}\in\Omega_{D_M}} \left\| I_M(\mathbf{x}) - I(\omega(\mathbf{x}, D_M(\mathbf{x}), \boldsymbol{\xi})) \right\|_\delta , \tag{5.1}$$

where $\omega\colon \Omega_{D_M} \times \mathbb{R} \times \mathrm{SE}(3) \to \Omega$ projects a point from the reference image into the new frame, and $\|\cdot\|_\delta$ is the Huber norm to account for outliers. Global brightness changes due to auto-shutter typically have little effect, as we only use image re-

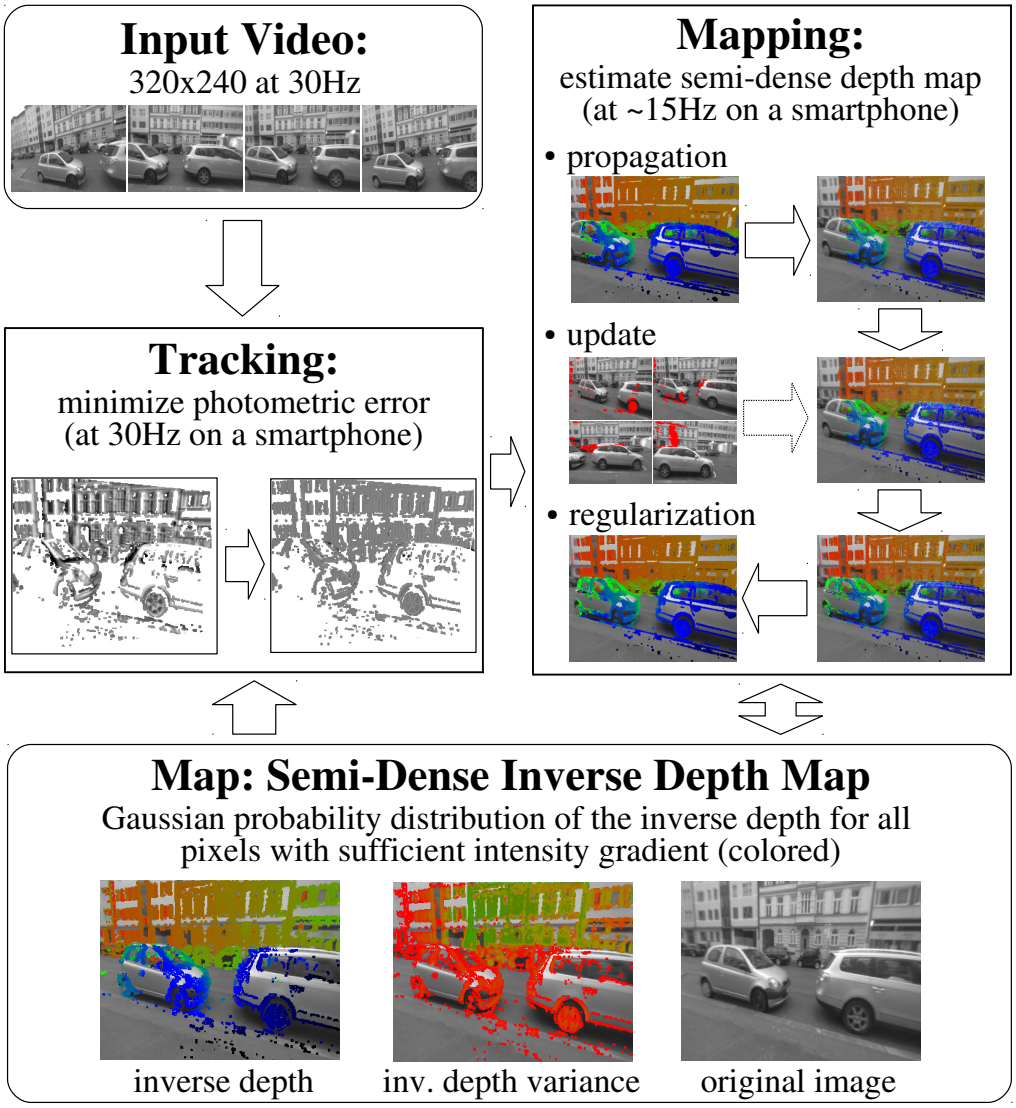


Figure 5.4: **Semi-Dense Visual Odometry.** Tracking and mapping are performed in parallel, operating on a semi-dense inverse depth map as central data structure. It contains an inverse depth hypothesis for all pixels close to sufficiently strong image gradient. It is continuously propagated to new frames, updated with new stereo observations and spatially regularized. More details on the respective steps are given in Sec. 5.2.

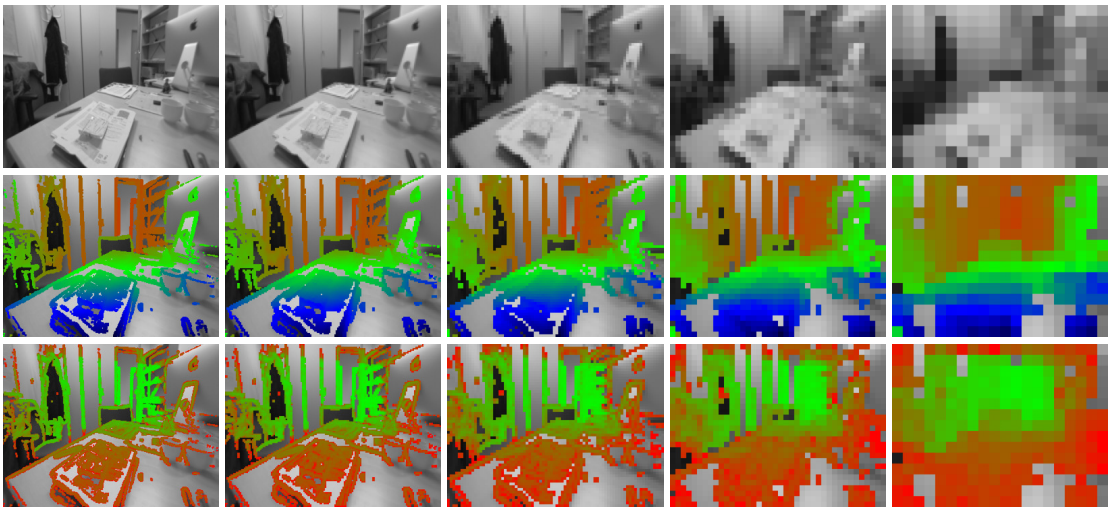


Figure 5.5: **Image Pyramid.** Top: intensity image I_l, middle: color-coded inverse depth D_l, bottom: inverse depth variance V_l. The given percentage corresponds to the density, i.e., the percentage of pixels that have a depth value. The described down-sampling strategy causes the depth maps to become significantly denser on higher pyramid levels.

gions with strong gradient. The minimum is computed using iteratively re-weighted Levenberg-Marquardt minimization, as described in [4].

Image Pyramid. To handle larger inter-frame motions, we use a pyramid scheme: Each new frame is first tracked on a very low resolution image and depth map, the tracked pose is then used as initialization for the next higher resolution. Depth maps are down-sampled by factors of two, using a weighted average of the *inverse* depth. To account for the strong correlation between neighbouring pixels, we average the information (inverse variance), giving

$$D_{l+1}(\mathbf{x}) := \frac{\sum_{\mathbf{x}' \in \Omega_\mathbf{x}} \frac{D_l(\mathbf{x}')}{V_l(\mathbf{x}')}}{\sum_{\mathbf{x}' \in \Omega_\mathbf{x}} \frac{1}{V_l(\mathbf{x}')}} \tag{5.2}$$

$$V_{l+1}(\mathbf{x}) := \frac{|\Omega_\mathbf{x}|}{\sum_{\mathbf{x}' \in \Omega_\mathbf{x}} \frac{1}{V_l(\mathbf{x}')}} \tag{5.3}$$

where l is the index of the pyramid level. $\Omega_\mathbf{x}$ denotes the set of valid pixels (i.e. with depth value) contained in pixel $\mathbf{x}$ at the next higher resolution.

Note that a pixel at low resolution has an associated depth hypothesis if at least one of the contained high-resolution pixels has a depth hypothesis. This strategy automatically lets the semi-dense depth maps become denser on lower resolutions (see Fig. 5.5), leading to more robust low-resolution tracking results without affecting the accuracy of the final result on the highest resolution. Averaging the inverse

depth effectively averages the optical flow, causing this strategy to work well for minimizing the photometric error (5.1). If used for reconstruction purposes however, it will create undesired points between the front and back surface around depth discontinuities.

5.2.2 Mapping

Depth maps are estimated by filtering over small-baseline pixel-wise stereo comparisons, interleaved with spatial regularization and propagation to a new frame, as first proposed in [9]. Each mapping iteration consists of three main steps:

1. **Propagation**: Depth hypotheses are projected into the most recently tracked frame, giving an initialization for the new depth map (prediction in an extended Kalman filter (EKF)).

2. **Update**: New depth measurements are obtained from a large number of pixel-wise stereo comparisons with previous frames, and merged into the existing depth map by filtering (observation in an EKF). We use the propagated prior hypothesis to constrain the search interval, greatly accelerating the search and reducing the probability for false observations in repetitive image regions.

   Stereo is only performed for a subset of suitable pixels, that is pixels where the expected accuracy is sufficiently high. This depends on the intensity gradient at that point as well as the camera motion, and is efficiently determined as proposed in [9]. In particular, regions with little image gradient are never updated as no accurate stereo measurements can be obtained. Ω_D contains all pixels that have a depth hypothesis, either propagated from previous frames, or observed in that frame.

3. **Regularization**: In a last step, the depth map is spatially regularized and outliers are removed.

Mapping runs in a continuous loop, in each iteration propagating the depth map to the most recently tracked frame, potentially skipping some frames. The runtime of one iteration varies in practice, as it depends on the density of the current depth map and the camera motion; an experimental runtime evaluation is given in Sec. 5.4.

5.2.3 Implementation on Mobile Phones

Current smartphone cameras have a rolling shutter, which introduces systematic distortions and, during quick motion, can have strong effects on the accuracy of stereo observations. While there exist methods to correctly model this in an off-line reconstruction setting [101], or to approximate it in real-time [62, 76], we found that ignoring the rolling shutter still gives very good results in practice, and significantly saves computational time.

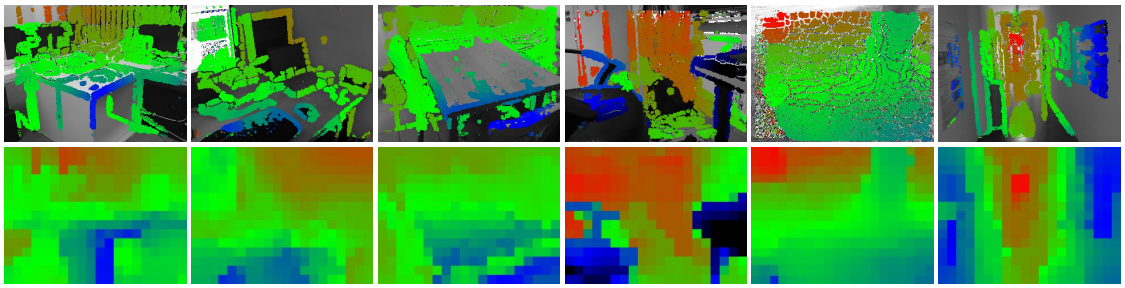


Figure 5.6: **Variational Inpainting.** The top row shows a number of full-resolution depth maps from live operation, the bottom row shows the computed low-resolution, regularized version used to build the collision mesh.

All experiments are conducted on a Sony Xperia Z1, which is equipped with a 2.3 GHz quad-core CPU. While the processing power of mobile devices has increased rapidly in the last years, mobile processors based on the ARM architecture are generally still much slower than their desktop counterparts; we currently do not use GPU or DSP features for tracking or mapping. In order to achieve real-time performance, i.e. tracking with at least 30 fps under these conditions, two steps were crucial: (1) separation of mapping and tracking resolution and choice of a suitable compromise, and (2) NEON optimization of computation-heavy algorithmic steps.

Image Resolution. While current desktop CPUs easily allow for real-time operation at VGA resolution (640×480), our mobile implementation performs mapping at 320×240. As tracking performance is crucial for a smooth AR experience, we further reduce the maximum resolution used for tracking down to 160×120. While this greatly reduces the computation time, the effect on accuracy is relatively small (see Sec. 5.4). This can be explained by the sub-pixel accuracy of direct image alignment: In practice, inaccuracies from motion blur, rolling shutter and other model violations (e.g. reflections, occlusions, specular highlights, etc.) dominate the error.

NEON Parallelization. Many parts of the tracking stage are well suited for optimization using SIMD parallelization. We use NEON instructions, which offer this functionality on ARM processors, leading to greatly improved performance and thereby being a vital step in achieving real-time performance on mobile processors. There are two algorithmic steps in tracking which particularly benefit from NEON optimization:

(1) Calculating the approximated Hessian $\mathbf{H}$ and gradient $\mathbf{g}$ of the error required for building the linear system to compute the pose increment $\boldsymbol{\Delta\xi}$, that is

$$\underbrace{\sum_{\mathbf{x}\in\Omega_D} \mathbf{J}_\mathbf{x}^T \mathbf{J}_\mathbf{x} w_\mathbf{x}}_{\mathbf{H}} \cdot \boldsymbol{\Delta\xi} = \underbrace{\sum_{\mathbf{x}\in\Omega_D} \mathbf{J}_\mathbf{x}^T r_\mathbf{x} w_\mathbf{x}}_{\mathbf{g}}, \tag{5.4}$$

where $r_\mathbf{x}$, $\mathbf{J}_\mathbf{x}$ and $w_\mathbf{x}$ are, for one pixel $\mathbf{x}$, the pixel's residual, its Jacobian with

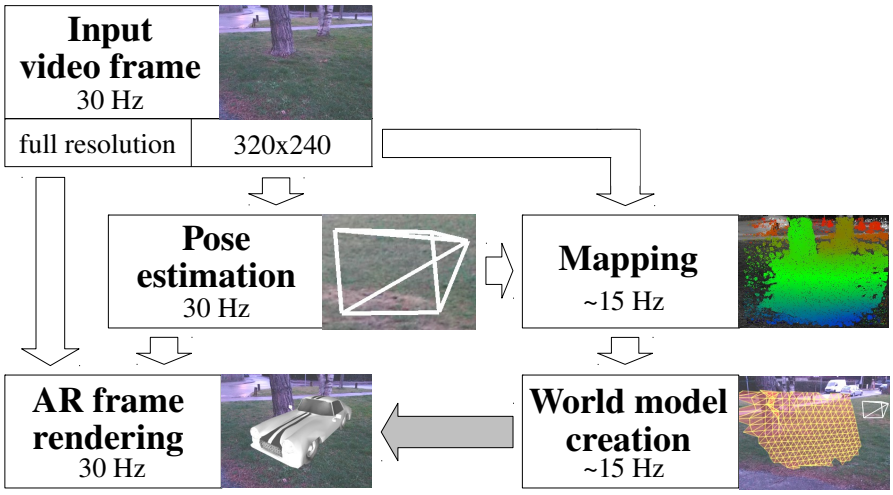


Figure 5.7: **AR Processing Pipeline.** Camera images are retrieved and displayed directly at full resolution. Simultaneously, a downsampled (320×240) version is computed and processed in the VO pipeline to allow real-time tracking and mapping. The estimated pose of that frame, as well as the generated world model are then used to render virtual objects into the scene. The world model is updated asynchronously at a lower frequency. All components run in parallel.

respect to the pose update, and the computed Huber weight. Using NEON optimization, four elements in the sum – which goes over all pixels which have depth – can be processed at once, resulting in a significant speed-up.

(2) Calculating the weights and residual sum: again, four pixels can be processed at the same time. In addition, NEON offers fast inverse approximations, which help to reduce processing time.

Both the above steps are required in every Levenberg-Marquardt iteration, thereby making up a large part of overall tracking performance. Details to the runtime of our implementation, with and without NEON acceleration, are given in Sec. 5.4.

5.3 Augmented Reality Application

We demonstrate a simple AR game using the computed semi-dense depth maps, in which a simulated car can be driven through the environment. For this, we construct a low-resolution collision mesh from the semi-dense depth map, which is used for real-time physics simulation with the free Bullet library [28].

For this, we assume that the scene has a well-defined ground plane, which we estimate with the help of the IMU. The full processing pipeline for augmented reality is shown in Fig. 5.7.

5.3.1 Collision Mesh Generation

We first compute a fully dense low-resolution (15×20) depth map using a variational in-painting approach. As data term for valid pixels we use the hypothesis from the corresponding level of the semi-dense depth map. Additionally, to cover up large unconstrained regions, we assume that pixels that do not have a depth hypothesis lie on the estimated ground plane π. As regularizer we use the Huber norm

$$\|\mathbf{x}\|_\delta := \begin{cases} \frac{\|\mathbf{x}\|_2^2}{2\delta} & \text{if } ||\mathbf{x}||_2 < \delta \\ \|\mathbf{x}\|_1 - \frac{\delta}{2} & \text{otherwise} \end{cases} \tag{5.5}$$

of the inverse depth gradient. The Huber norm is a combination of a quadratic regularizer favouring smooth surfaces, and the total variation (TV), which allows sharp transitions at occluding edges. The combined energy to be minimized with respect to the resulting inverse depth map u is hence given by

$$\begin{aligned} E(u) := & \int_{\Omega_D} \frac{(u(\mathbf{x}) - D(\mathbf{x}))^2}{V(\mathbf{x})} d\mathbf{x} \\ & + \int_{\Omega \setminus \Omega_D} \frac{(u(\mathbf{x}) - \pi(\mathbf{x}))^2}{V_\pi} d\mathbf{x} \\ & + \alpha \int_\Omega \|\nabla u\|_\delta d\mathbf{x}, \end{aligned} \tag{5.6}$$

where $\pi(\mathbf{x})$ denotes the inverse depth of pixel $\mathbf{x}$ assuming it lies on the estimated ground plane, while V_π and α are parameters of the energy functional. This is a convex energy, and on the used resolution can be minimized globally and quickly using gradient descent. Fig. 5.6 shows some results. Afterwards, a triangle mesh is generated from the resulting depth map by interpreting the depth pixels as corners of a regular triangle grid. Some examples in different scenes are shown in Fig. 5.9. A desirable effect of this approach is that the collision meshes naturally have a higher resolution in close-by than in far-away regions, where the un-projected mesh vertices are more tightly spaced and at the same time the depth map is more accurate.

5.3.2 Ground Plane Estimation

We estimate the ground plane normal by low-pass filtering accelerometer measurements which are available on all modern smartphones, giving the direction of gravity. To determine the plane height, we search for the lowest height which is supported by a certain minimum number of depth map samples. The maximum height of all supporting samples is then taken as ground plane: this assures that small bumps, caused by inaccurate height estimates of individual samples, are covered up with a smooth ground surface to drive on.

method	mapping	tracking	fr2/xyz (cm/s)	fr2/xyz (deg/s)	fr2/desk (cm/s)	fr2/desk (deg/s)
PTAM	640×480	640×480	8.2	3.2	fail	fail
ours	640×480	640×480	0.50	0.31	2.2	0.96
ours	640×480	320×240	0.58	0.32	3.6	1.25
ours	320×240	320×240	0.58	0.32	3.3	0.96
ours	320×240	160×120	0.62	0.33	4.9	1.38
ours	160×120	160×120	0.68	0.37	fail	fail
ours	160×120	80×60	1.58	0.71	fail	fail

Table 5.1: **Accuracy Evaluation.** Tracking accuracy (as drift per second) for two sequences from the TUM RGB-D benchmark [114] at different resolutions. For comparison, we also include results obtained from PTAM [69]. The sequences contain significant rolling shutter and motion blur effects, which reduce the tracking accuracy of PTAM significantly. Accuracy of direct tracking degrades only very little with decreasing image resolution.

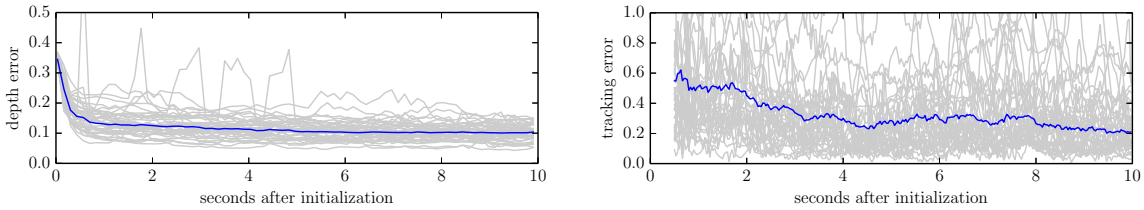


Figure 5.8: **Initialization evaluation.** Development of errors in successful random initialization runs (average in blue, all samples in gray), evaluated on subsequences of the TUM RGB-D benchmark fr2/desk sequence. The depth error shows the relative deviation from the ground truth depth, after choosing the optimal scale. The tracking error shows the translational drift per 0.5 seconds, relative to the traveled distance. The largest reduction in depth error happens already within the first 0.5 seconds; the tracking error requires longer to stabilize.

5.4 Results

Initialization. We evaluate the success rate of random initialization by running our system on many subsequences of the fr2/desk sequence of the TUM RGB-D benchmark [114]. Note that this includes subsequences with all types of motion, in particular strong rotation or forward-translation, which are ill-conditioned for initialization – causing the initialization to fail more often than in a hand-held case.

A run is classified as successful if the mean relative depth error after 3 seconds is at most 16%, and final relative translational drift (computed over 15 frames) is at most 60%. We observed that the success rate strongly depends on the movement speed of the camera: if the camera does not move sufficiently fast, the depth filters become over-confident, and get stuck at wrong values – this however can be avoided by only mapping on a subset of frames, e.g. every 4th frame. Overall, the measured

	C++ 320×240	ASM with NEON 640×480	ASM with NEON 320×240
Tracking	30.7 (±11.2)	39.2 (±15.9)	14.7 (±6.8)
Mapping	46.6 (±35.4)	184.4 (±123.3)	52.6 (±37.7)

Table 5.2: **Computation Speed Evaluation.** Performance of tracking and mapping for the first 600 frames of the fr2/desk sequence, on a Sony Xperia Z1 (mean and standard deviation, in ms per iteration). The given resolution is the mapping resolution, tracking is done with one pyramid level higher as the last level. Mapping is not NEON-optimized; the frames were played back with 30 fps. Note that tracking and mapping run in parallel.

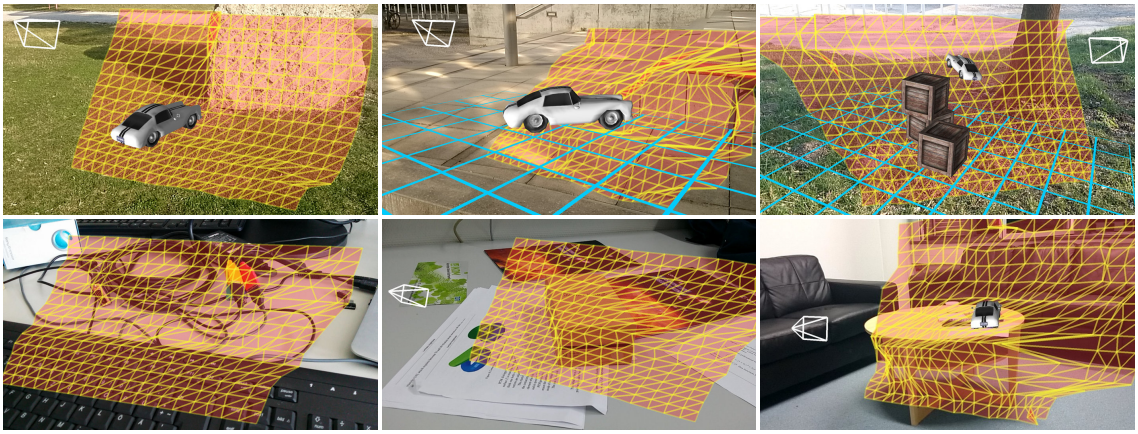


Figure 5.9: **Example Images.** Qualitative evaluation of the system in different, challenging scenes. The collision mesh is fixed and shown from a different perspective, together with its original viewport, augmented objects and the ground plane. The screenshots are taken by the smartphone during live operation.

initialization success rate with this configuration is 67%. Figure 5.8 shows the evolution of the relative translational drift as well as the mean relative depth error over the first 10 s of all successful runs. Note that these results are obtained using resolutions as employed on the smartphone, and some of the sequences contain strong motion blur and rolling shutter artifacts.

Accuracy. We numerically evaluate the tracking accuracy of the proposed approach for different resolutions using the TUM RGB-D benchmark. To be independent from initialization issues and to obtain the correct scale, we use the very first depth image for initialization, while for the remainder of the sequences only the provided intensity images are used. Table 5.1 shows the results. Notably, the accuracy only changes very little with decreasing image resolution, allowing smooth yet accurate operation on a smartphone.

Speed. With NEON optimizations at a resolution of 320×240, our system is able to track the camera pose with usually well more than 30 Hz on current-generation

smartphones. See Table 5.2 for timing values measured on a Sony Xperia Z1.

Qualitative Results. We extensively tested the system in real-time operation, Fig. 5.9 shows some examples of augmented scenes. A full sequence is shown in the attached video.

5.5 Conclusion

The presented direct monocular visual odometry algorithm is able to operate in real-time on a modern smartphone, with tracking rates of well above 30 Hz at a mapping resolution of 320 × 240. It operates fully without features; instead it is based on direct image alignment for tracking, and semi-dense depth estimation by pixel-wise filtering over many small-baseline stereo comparisons for mapping. This allows to use much more information in the images (including e.g. edges) and reduces the number of outliers drastically. In addition to accurately and robustly estimating the camera pose, the estimated semi-dense depth maps can be used to build a physical world model for AR with little additional computational effort. We demonstrated this with a small example application.

As future work, using a more sophisticated regularizer for depth map in-painting (e.g. total generalized variation) will eliminate the need for estimating a ground plane. At the same time, more sophisticated minimization schemes as e.g. in [112] will allow world modeling on higher resolutions. Integrating the recent extension towards full, large-scale direct monocular SLAM (LSD-SLAM) [4] will allow to merge collision meshes and scale the method to larger environments.

Chapter 6

Large-Scale Direct SLAM with Stereo Cameras

Authors	Jakob Engel[1]	engelj@in.tum.de
	Jörg Stückler[2]	stueckler@vision.rwth-aachen.de
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] RWTH Aachen University	

Publication **Large-Scale Direct SLAM with Stereo Cameras**. J. Engel, J. Stückler, D. Cremers. In *Preceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), 2015.*

doi: 10.1109/IROS.2015.7353631

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

Abstract. We propose a novel Large-Scale Direct SLAM algorithm for stereo cameras (Stereo LSD-SLAM) that runs in real-time at high frame rate on standard CPUs. In contrast to sparse interest-point based methods, our approach aligns images directly based on the photoconsistency of all high-contrast pixels, including corners, edges and high texture areas. It concurrently estimates the depth at these pixels from two types of stereo cues: Static stereo through the fixed-baseline stereo camera setup as well as temporal multi-view stereo exploiting the camera motion. By incorporating both disparity sources, our algorithm can even estimate depth of pixels that are under-constrained when only using fixed-baseline stereo. Using a fixed baseline, on the other hand, avoids scale-drift that typically occurs in pure monocular SLAM. We furthermore propose a robust approach to enforce illumination invariance, capable of handling aggressive brightness changes between frames – greatly improving the performance in realistic settings. In experiments, we demonstrate state-of-the-art results on stereo SLAM benchmarks such as Kitti or challenging datasets from the EuRoC Challenge 3 for micro aerial vehicles.

6.1 Introduction

Visual simultaneous localization and mapping (SLAM) under real-time constraints has traditionally been tackled using sparse interest points, since they reduce the large amount of pixels in images to a small amount of features. Only recently, real-time capable direct methods have been proposed that avoid the reliance on interest points, but instead perform image alignment and 3D reconstruction directly on pixels using photoconsistency constraints. The premise of direct approaches over interest-point based methods is that image information can be used densely. No manual design of interest point detectors, descriptors, and matching procedures is required, which would also restrict the SLAM algorithm to a specific type of feature – typically only image corners are used. Instead in direct SLAM methods, a rich set of pixels contributes to depth estimation and mapping.

In this paper, we propose the first large-scale direct visual SLAM approach for stereo cameras that is real-time capable on CPUs. Our method estimates depth with uncertainty estimates at pixels with high intensity gradient, reconstructing a semi-dense depth map online. It concurrently tracks the rigid-body motion through photometric alignment of images based on the depth maps.

In our previous work on large-scale direct monocular SLAM (LSD-SLAM), we obtain depth in keyframes by pixel-wise stereo between the current and the keyframe. Camera motion is tracked towards a keyframe through photometric image alignment. For SLAM on the global scale, keyframes are aligned towards each other and their poses are optimized by graph optimization. Since reconstruction scale is

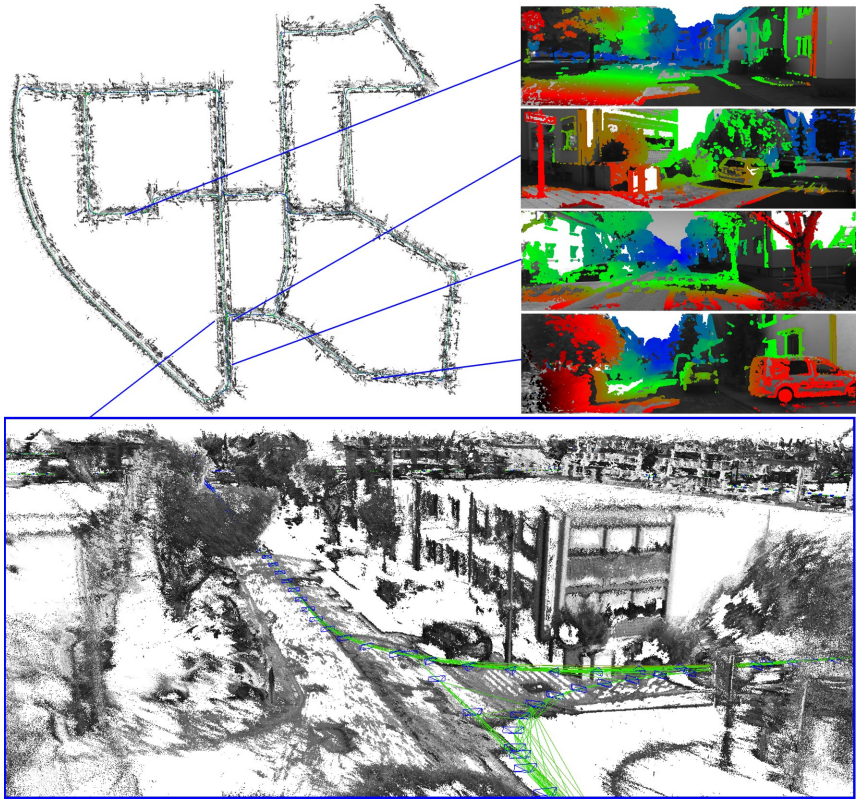


Figure 6.1: **KITTI Reconstruction Example.** Stereo LSD-SLAM is a fully direct SLAM method for stereo cameras. It runs at 30Hz on a CPU, computing accurate camera movement as well as semi-dense probabilistic depth maps. We exploit both static and temporal stereo and correct for affine lightning changes, making the method both accurate and robust in real-world scenarios. Some examples are shown in the attached video.

not observable in monocular SLAM, we additionally optimize for the scale in direct image alignment as well as in pose graph optimization.

In this work, we couple *temporal stereo* of monocular LSD-SLAM with *static stereo* from a fixed-baseline stereo camera setup. At each pixel, our Stereo LSD-SLAM method integrates static as well as temporal stereo cues into the estimate depending on availability. This combines the properties of monocular structure from motion with fixed-baseline stereo depth estimation in a single SLAM method. While static stereo effectively removes scale as a free parameter, temporal stereo cues allow for estimating the depth from baselines beyond the small baseline of the stereo camera. Temporal stereo is not restricted to one specific (e.g. horizontal) direction like static stereo. Rather its baseline corresponds to the translational motion between frames. We furthermore propose a method for handling illumination changes in direct image alignment which significantly improves the robustness of our algorithm in realistic settings.

We evaluate Stereo LSD-SLAM on the popular Kitti benchmark and datasets

from the EuRoC Challenge 3 for micro aerial vehicles (MAVs), demonstrating the state-of-the-art performance of our approach.

6.2 Related Work

Sparse interest-point-based approaches to visual odometry and SLAM have been extensively investigated in recent years. The term visual odometry has been coined in the seminal work of Nister et al. [89] who proposed sparse methods for estimating the motion of monocular as well as stereo cameras by sequential frame-to-frame matching. Chiuso et al. [23] proposed one of the first real-time capable monocular SLAM methods based on non-linear filtering. Davison [31] proposed MonoSLAM, a real-time capable, EKF-based method that demonstrated SLAM in small workspaces. Sparse interest points are tracked in an EKF-SLAM formulation in order to recover camera motion and the (global) 3D position of the interest points. Another example of sparse monocular SLAM is Parallel Tracking and Mapping (PTAM [69]) which separates and parallelizes optimization for tracking and mapping in a bundle adjustment framework. More recently, Strasdat et al. [109] included scale as a parameter in a key-frame-based optimization approach to sparse monocular SLAM.

Using a fixed-baseline stereo camera setup, scale becomes directly observable. One early work applies EKF-SLAM on a sparse set of interest points [30]. Paz et al. [92] combine monocular stereo cues with fixed-baseline stereo in a sparse hierarchical EKF-SLAM framework.

Direct methods that avoid the detection of sparse interest points have recently attracted attention for visual SLAM. One major advantage of direct over sparse methods is that they do not rely on manually designed image features which constrain the type of information that can be used in subsequent processing stages. In the RGB-D domain [65, 66, 83], direct methods have become the state-of-the-art for their high accuracy and efficiency. LSD-SLAM [4] has been the first large-scale direct monocular SLAM method. In LSD-SLAM, camera motion is tracked towards keyframes for which semi-dense depth maps are estimated using probabilistic filtering. Pose graph optimization aligns the keyframes in a globally consistent arrangement. LSD-SLAM explicity considers scale drift in pose graph optimization and finds a single consistent scale. For stereo cameras, a direct visual odometry approach has been proposed by Comport et al. [26]. Their approach does not explicitly recover depth, but uses quadrifocal constraints on pixels which are in stereo correspondence for camera motion estimation. In the direct stereo method in [118], a disparity map is integrated over time, while the motion of the stereo camera is tracked through direct image alignment using the estimated depth. The keyframes in our approach also integrate depth, while we employ probabilistic filtering instead. Our approach combines fixed-baseline stereo cues from the static camera setup with temporal stereo from varying baselines caused by the moving camera. We combine

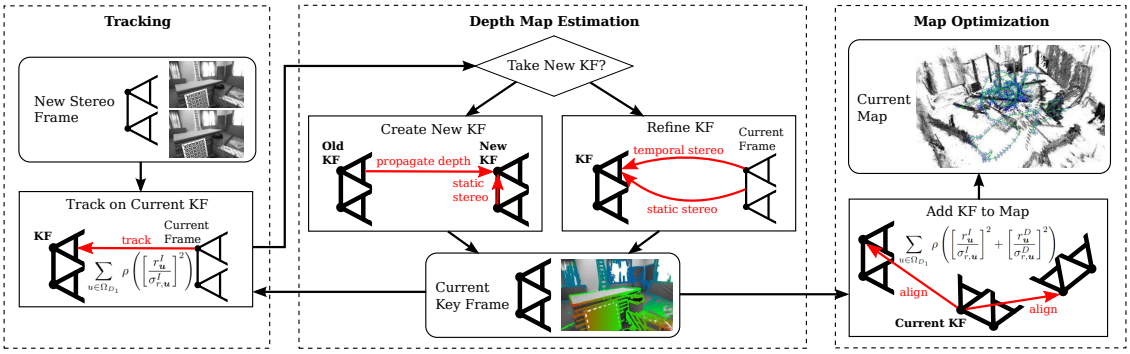


Figure 6.2: **Overview.** Schematic layout of the Stereo LSD-SLAM system.

this with a pose-graph-based SLAM system that globally optimizes the poses of the keyframes. A further important contribution of our work is the correction for affine lighting changes to enable direct image alignment in realistic settings. Differently to previous methods [49, 71], we optimize for affine lighting correction parameters in an alternating fashion, which allows for different outlier rejections schemes to be applied in image alignment and lighting correction.

6.3 LSD-SLAM with Stereo Cameras

LSD-SLAM [4] is a key-frame based localization and mapping approach which uses the following main steps:

- The motion of the camera is tracked towards a reference keyframe in the map. New keyframes are generated if the camera moved too far from existing keyframes in the map.
- Depth in the current reference keyframe is estimated from stereo correspondences based on the tracked motion (temporal stereo).
- The poses of the keyframes are made globally consistent by mutual direct image alignment and pose graph optimization.

In Stereo LSD-SLAM, the depth in keyframes is in addition directly estimated from static stereo (see Fig. 6.2). There is a number of advantages of this approach to relying solely on temporal or solely on static stereo. Static stereo allows for estimating the absolute scale of the world and is independent of the camera movement. However, static stereo is constrained to a constant baseline (with, in many cases, a fixed direction), which effectively limits the performance to a specific range. Temporal stereo does not limit the performance to a specific range as demonstrated in [4]. The same sensor can be used in very small and very large environments, and seamlessly transits between the two. On the other hand, it does not provide scale and requires non-degenerate camera movement. An additional benefit of combining

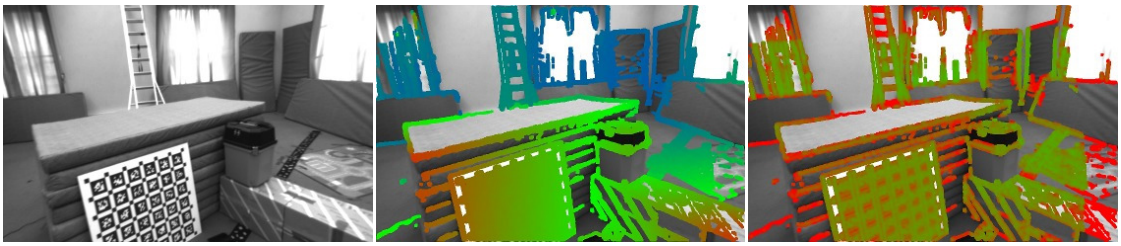


Figure 6.3: **Geometry Representation.** Each keyframe maintains a Gaussian probability distribution on the inverse depth for all pixels that have sufficient image gradient such that the depth can be estimated. From left to right: Intensity image, semi-dense inverse depth map, inverse depth variance map.

temporal and static stereo is, that 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.

In detail, we make the following key contributions:

- We generalize LSD-SLAM to stereo cameras, combining temporal and static stereo in a direct, real-time capable SLAM method.
- We explicitly model illumination changes during direct image alignment, thereby making the method highly robust even in challenging real-world conditions.
- We perform a systematic evaluation on two benchmark datasets from realistic robotics applications, demonstrating the state-of-the-art performance of our approach.

6.3.1 Notation

We use bold capital letters for matrices (such as $\mathbf{R}$) and bold lower case letter for vectors (such as $\boldsymbol{\xi}$). The operator $[\cdot]_n$ selects the n-th row of a matrix. Throughout the paper we use d to denote the *inverse* of the depth z of a point, i.e., $d = z^{-1}$.

In Stereo LSD-SLAM, a map is maintained as a set of keyframes $\mathcal{K}_i = \left\{I_i^l, I_i^r, D_i, V_i\right\}$. Each keyframe consists of the left and right image $I_i^{l/r} \colon \Omega \to \mathbb{R}$ of the stereo camera, an inverse depth map $D_i \colon \Omega_{D_i} \to \mathbb{R}^+$ and its variance map $V_i \colon \Omega_{D_i} \to \mathbb{R}^+$. Depth and variance are only maintained for one of the images in the stereo pair, we always use the left image as reference frame. We assume the image domain $\Omega \subset \mathbb{R}^2$ to be given in stereo-rectified image coordinates, i.e., the intrinsic and extrinsic camera parameters are known a-priori. The domain $\Omega_{D_i} \subset \Omega$ is the semi-dense restriction to the pixels which are selected for depth estimation.

We denote pixel coordinates by $\boldsymbol{u} = (u_x\, u_y\, 1)^T$. A 3D position $\boldsymbol{p} = (p_x\, p_y\, p_z\, 1)^T$ is projected into the image plane through the mapping

$\boldsymbol{u} = \pi(\boldsymbol{p}) := \mathbf{K}\left((p_x/p_z)\,(p_y/p_z)\,1\right)^T$, where $\mathbf{K}$ is the camera matrix. The mapping $\boldsymbol{p} = \pi^{-1}(\boldsymbol{u}, d) := \left((d^{-1}\mathbf{K}^{-1}\boldsymbol{u})^T\,1\right)^T$ inverts the projection with the inverse depth d.

6.3.2 Depth Estimation

We estimate the geometry of the scene in keyframes. Each keyframe maintains Gaussian probability distributions on the inverse depth of a subset of pixels. This subset is chosen as the pixels with high image gradient magnitude, since these pixels provide rich structural information and more robust disparity estimates than pixels in textureless areas. Figure 6.3 shows an example of such a semi-dense depth map and associated variance map. We initialize the depth map by propagating depth hypothesis 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.

In contrast to monocular SLAM, depth is estimated 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).

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.

As correspondence measure we use the SSD photometric error over five pixels along the scanline. After subpixel accurate refinement of the disparity, its variance is estimated through the geometric and photometric error identified in [9]. If a Gaussian prior with mean d and standard deviation σ_d on the inverse depth is available, we constrain the search to $[d - 2\sigma_d, d + 2\sigma_d]$. In practice, the search interval consists of only very few pixels for all but newly initialized hypothesis, greatly accelerating the search and reducing the probability of finding an incorrect or ambiguous match. According to the two error sources, we expect that pixels with image gradients close to vertical, or with low image gradient along the horizontal direction do not provide accurate disparity estimates. Hence, we neglect these pixels for static stereo.

When a new keyframe is initialized, we immediately perform static stereo to update and prune the propagated depth map. In particular, pruning removes pixels that became occluded, and we fill in holes arising from forward-warping the depth map. Subsequently, we also make use of static stereo from tracked non-keyframes, and integrate the obtained disparity information into the keyframe they were tracked on: In a first step, the inverse depth hypothesis at a pixel $\boldsymbol{u}$ in the keyframe is

transformed into the new frame,

$$\boldsymbol{u}' = \pi\left(\mathbf{T}_{\boldsymbol{\xi}}\pi^{-1}\left(\boldsymbol{u}, d\right)\right) \tag{6.1}$$

$$d' = \left[\mathbf{T}_{\boldsymbol{\xi}}\pi^{-1}\left(\boldsymbol{u}, d\right)\right]_3^{-1} \tag{6.2}$$

$$\sigma_{d'}^2 = \left(\frac{d}{d'}\right)^4 \sigma_d^2, \tag{6.3}$$

according to the pose estimate $\boldsymbol{\xi}$. The propagated hypothesis is used as prior for a stereo search, and the respective observed depth d'_{obs} and observation variance $\sigma^2_{d',\text{obs}}$ is determined. Finally, the observation is transformed back into the keyframe using

$$d_{\text{obs}} = \left[\mathbf{T}_{\boldsymbol{\xi}}^{-1}(\pi^{-1}\left(\boldsymbol{u}', d'_{\text{obs}}\right))\right]_3^{-1} \tag{6.4}$$

$$\sigma^2_{\text{obs}} = \left(\frac{d'_{\text{obs}}}{d_{\text{obs}}}\right)^4 \sigma^2_{d',\text{obs}}, \tag{6.5}$$

and fused into the depth map. Note that observations from non-keyframes can only be generated for pixels with an existing prior hypothesis – new hypothesis are only generated during stereo on the keyframe, or from temporal stereo. This process is schematically shown in Fig. 6.2.

Temporal Stereo After tracking, we estimate disparity between the current frame and the reference keyframe and fuse it in the keyframe. Again, we only use pixels for which the expected inverse depth error is sufficiently small. We determine this uncertainty from several criteria: the image gradient should be sufficiently large, not be parallel to the epipolar line and the pixel should not be close to the epipole. We kindly refer to [9] for further details on this method. While we use a simple 5-pixel SSD error, we correct for affine lighting changes with the affine mapping found during tracking, as will be described in Sec. 6.3.3. Note that for temporal stereo, the geometric error typically is higher than for static stereo, as relative camera pose stems from direct image alignment. This pose estimate often is less accurate than the offline calibrated extrinsic calibration between the stereo camera pair.

6.3.3 Direct Image Alignment with Affine Lighting Correction

We determine the camera motion between two images using direct image alignment. We use this method to track camera motion towards a reference keyframe. It is also used for estimating relative pose constraints between keyframes for pose graph optimization. Finally, we propose a robust method to compensate for affine lighting changes.

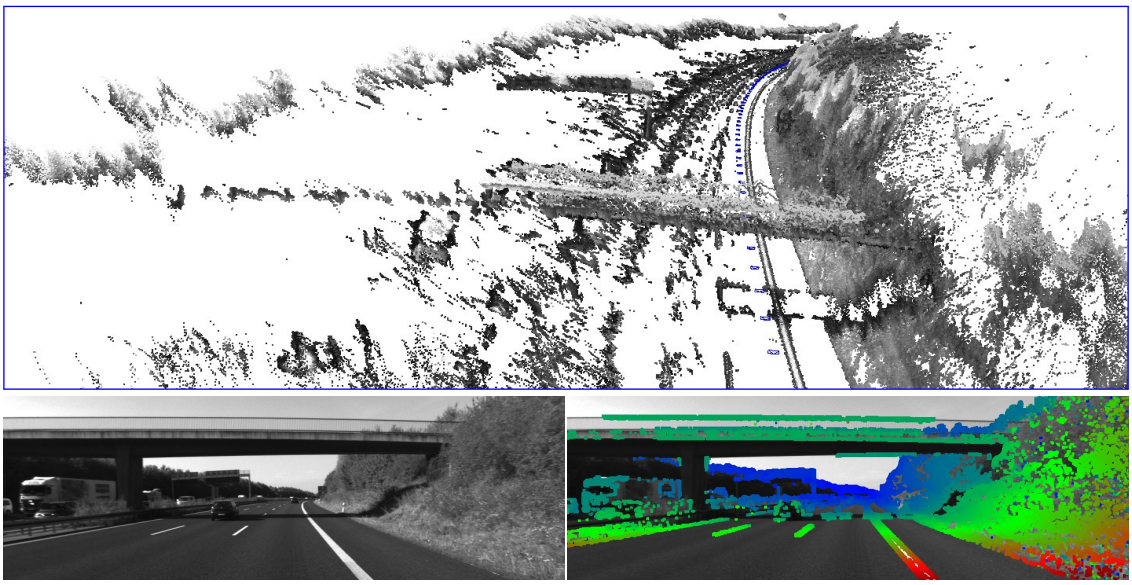


Figure 6.4: **Temporal vs. Static Stereo.** Example of a scene where both temporal stereo (epipolar lines are parallel to the lane-markings on the road) and static stereo (epipolar lines are parallel to the horizontal bridge) alone fail to capture all information present. Our combined approach fuses information from both, and hence can reconstruct everything in the scene.

Direct Image Alignment

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

$$r_{\boldsymbol{u}}^I(\boldsymbol{\xi}) := I_1^l(\boldsymbol{u}) - I_2^l(\pi(\boldsymbol{p}')) \tag{6.6}$$

where $\boldsymbol{p}' := \mathbf{T}_{\boldsymbol{\xi}}\pi^{-1}(\boldsymbol{u}, D_1(\boldsymbol{u}))$ and $\boldsymbol{\xi}$ transforms from image frame I_2^l to I_1^l. We also determine the uncertainty $\sigma_{r,\boldsymbol{u}}^I$ of this residual [4]. The optimization objective for tracking a current frame towards a keyframe is

$$E^{\text{track}}(\boldsymbol{\xi}) := \sum_{u \in \Omega_{D_1}} \rho\left(\frac{r_{\boldsymbol{u}}^I(\boldsymbol{\xi})}{\sigma_{r,\boldsymbol{u}}^I}\right), \tag{6.7}$$

where ρ is a robust weighting function; we choose ρ as the Huber norm. Note that in contrast to [26], we only align I_1^l to I_2^l. While one could choose to add photometric constraints to the new right image I_2^r, we observed that this can decrease accuracy in practice: typically, the baseline from I_1^l to I_2^r is much larger than to I_2^l, leading to more outliers from occlusions and reflections.

Since fused depth is available in keyframes, we add geometric residuals for keyframe-to-keyframe alignment,

$$r_{\boldsymbol{u}}^D(\boldsymbol{\xi}) := [\boldsymbol{p}']_3 - D_2(\pi(\boldsymbol{p}')) \tag{6.8}$$

providing additional information that is not available when initially tracking new frames, since these not have associated depth estimates yet. The combined objective is

$$E^{\text{keyframes}}(\boldsymbol{\xi}) := \sum_{u \in \Omega_{D_1}} \left[\rho\left(\frac{r^I_{\boldsymbol{u}}(\boldsymbol{\xi})}{\sigma^I_{r,\boldsymbol{u}}} \right) + \rho\left(\frac{r^D_{\boldsymbol{u}}(\boldsymbol{\xi})}{\sigma^D_{r,\boldsymbol{u}}} \right) \right] \tag{6.9}$$

Note that this formulation exploits the full depth information available for both frames, including propagated and fused observations from other stereo pairs (see Sec. 6.3.2). This is in contrast to an implicit quadrifocal approach as e.g. in [26].

We minimize these objectives using the iteratively re-weighted Levenberg-Marquardt algorithm in a left-compositional formulation: Starting with an initial estimate $\boldsymbol{\xi}^{(0)}$, in each iteration a left-multiplied increment $\delta\boldsymbol{\xi}^{(n)}$ is computed by solving for the minimum of a second-order approximation of E, with fixed weights:

$$\delta\boldsymbol{\xi}^{(n)} = -(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda \text{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{6.10}$$

where

$$\mathbf{J} = \left. \frac{\partial \mathbf{r}(\boldsymbol{\epsilon} \circ \boldsymbol{\xi}^{(n)})}{\partial \boldsymbol{\epsilon}} \right|_{\boldsymbol{\epsilon}=0} \tag{6.11}$$

is the derivative of the stacked vector of residuals $\mathbf{r}(\boldsymbol{\xi})$ with respect to a left-multiplied increment $\boldsymbol{\epsilon}$, $\mathbf{J}^T\mathbf{W}\mathbf{J}$ the Gauss-Newton approximation of the Hessian of E, and $\mathbf{W}$ a diagonal matrix containing the weights. The new estimate is then obtained by multiplication with the computed update

$$\boldsymbol{\xi}^{(n+1)} = \delta\boldsymbol{\xi}^{(n)} \circ \boldsymbol{\xi}^{(n)}. \tag{6.12}$$

We use a coarse-to-fine scheme to improve efficiency and basin of convergence of the optimization.

Assuming the residuals to be statistically independent, the inverse of the Hessian from the last iteration $(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}$ is an estimate for the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ of a left-multiplied increment $\boldsymbol{\epsilon}$ onto the final minimum, that is

$$\boldsymbol{\xi}^{(n)} = \boldsymbol{\epsilon} \circ \boldsymbol{\xi}_{\text{true}} \quad \text{with} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}}). \tag{6.13}$$

In practice, the residuals are highly correlated, such that $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ is only a lower bound - yet it contains valuable information about the correlation between noise on the different degrees of freedom.

Affine Lighting Correction

Direct image alignment is fundamentally based on the brightness constancy assumption, which is heavily violated e.g. when the cameras exposure time is adjusted to

better fit the average brightness of the scene. A well-known countermeasure is to use a cost function that is invariant to *affine lighting changes*, e.g. using the normalized cross correlation (NCC) instead of a simple sum of squared differences (SSD) for matching. Here, we propose a similar approach, and modify the photometric residuals (6.6) to be invariant to affine lighting changes:

$$r_{\boldsymbol{u}}^{I}(\boldsymbol{\xi}) := aI_1^l(\boldsymbol{u}) + b - I_2^l(\boldsymbol{p}'). \tag{6.14}$$

Instead of a joint optimization for a, b and $\boldsymbol{\xi}$ in a common error formulation, we alternate between (1) a single Levenberg-Marquardt update step in $\boldsymbol{\xi}$ (fixing a, b) and (2) a full minimization over a, b (fixing $\boldsymbol{\xi}$), using different weighting schemes. This is motivated by the observation that $\boldsymbol{\xi}$ and a, b react very differently to outliers:

- The minimum in a, b is heavily affected by occluded and over-exposed pixels, as these tend to "pull" in the same wrong direction. On the other hand, it typically is well-constrained already by only a small number of inlier-residuals – we therefore employ a simple, aggressive cut-off SSD error, i.e., $\rho_{a,b}(r) := \min\{\delta_{\max}, r^2\}$. Fig. 6.5 shows two example scenes, and the resulting affine mapping with and without outlier rejection.
- The minimum in $\boldsymbol{\xi}$ is much less affected by outliers, as they tend to "pull" in different directions, cancelling each other out. In turn, it may happen that some dimensions of $\boldsymbol{\xi}$ are only constrained by a small amount of pixels, which initially have a high residual – removing these as outliers will cause the estimate to converge to a wrong local minimum. We therefore employ the weighting scheme proposed in [4], which only down-weights but does not remove residuals.

Minimization in a, b is done by iteratively minimizing

$$E_{a,b}(a,b) := \sum_{\boldsymbol{u}\in\Omega_{D_1}} \rho_{a,b}\left(\left(aI_1^l(\boldsymbol{u}) + b\right) - I_2^l(\boldsymbol{u}')\right) \tag{6.15}$$

with $\boldsymbol{u}' := \pi\left(\boldsymbol{p}'\right)$, which can be done in closed-form:

$$a^* = \frac{\sum_{\boldsymbol{u}\in\Omega_L} I_1^l(\boldsymbol{u}) I_2^l(\boldsymbol{u}')}{\sum_{\boldsymbol{u}\in\Omega_L} I_2^l(\boldsymbol{u}') I_2^l(\boldsymbol{u}')} \tag{6.16}$$

$$b^* = \frac{1}{|\Omega_L|} \sum_i \left(I_1^l(\boldsymbol{u}') - a^* I_2^l(\boldsymbol{u})\right), \tag{6.17}$$

with the set of inliers

$$\Omega_L := \left\{\boldsymbol{u} \in \Omega_{D_1} \mid \rho_{a,b}\left(\left(aI_1^l(\boldsymbol{u}) + b\right) - I_2^l(\boldsymbol{u}')\right) < \delta_{\max}\right\}.$$

The found affine parameters a, b are then used during temporal stereo and during the consistency check on depth propagation.

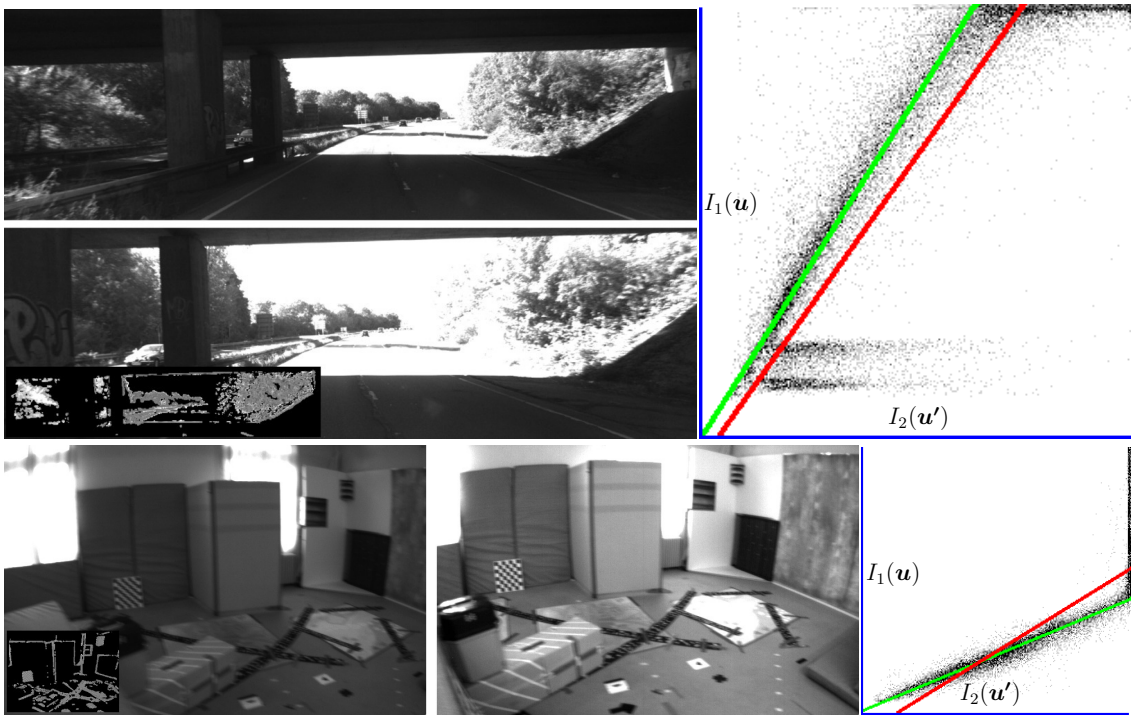
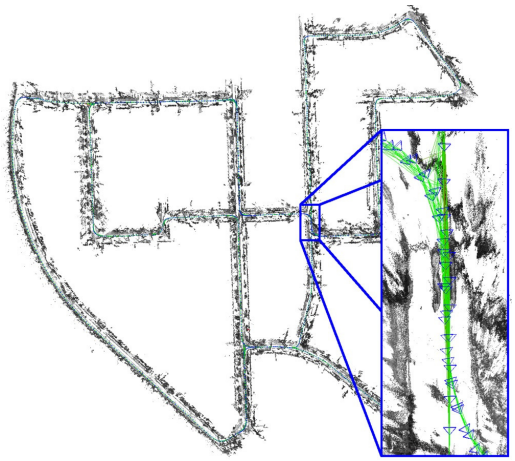


Figure 6.5: **Affine Lighting Correction.** Two scenes with strong lighting changes. On the right, we show a the scatter-plot of all residuals *after* direct image alignment; The green line shows the best fit from our approach, while the red line shows the best fit for all pixel. Note how it is heavily affected by outliers caused by occlusions and over-exposed pixels, which are easily recognizable in the scatter-plot.



Level	Num	Time
46×30	10.5k	0.6 ms
92×60	6.1k	2 ms
184×120	4.6k	8 ms
368×240	4.1k	24 ms

Figure 6.6: **Pyramid Tracking Speed.** Resulting pose-graph for Sequence 00 from the Kitti benchmark, containing 1227 keyframes and 3719 constraints. The table shows how many constraints have been attempted to track down to which pyramid level, as well as the average time required for reciprocal image alignment on that pyramid level. Note how most incorrect loop-closures candidates are discarded at very coarse resolution already, which is very fast. Over the whole sequence, only 43 large loop-closure attempts were required, to find all loop-closures in the sequence.

6.3.4 Key-Frame-Based SLAM

Once a keyframe $\mathcal{K}_i$ is finalized – that is, after it is replaced as tracking reference and will not receive any further depth updates – it is added to the pose-graph, which is continuously optimized in the background. Constraints are obtained by performing SE(3) alignment with depth residual and affine lighting correction to a set of possible loop-closure candidates: Tracking is attempted on all keyframes $\mathcal{K}_{j_1}, ..., \mathcal{K}_{j_n}$, which

- are at a physical distance of less than $(60 + p \cdot 0.05)$ m.
- have a difference in viewing direction of less than $(35 + p \cdot 0.01)°$.

where p is the length of the shortest connecting path in the keyframe graph between the two keyframes in meters, which serves as a conservative approximation to the accumulated relative pose error. For very large maps, additional loop-closures can be found by exploiting appearance-based image-retrieval techniques like FAB-MAP [29]. However in our experiments we did not find this to be necessary. For keyframes with $p \leq 100$ m, we use the relative pose obtained by composing edges along this path as initialization for direct image alignment, otherwise the identity is used.

For each candidate $\mathcal{K}_{j_k}$ we independently compute $\boldsymbol{\xi}_{j_k i}$ and $\boldsymbol{\xi}_{i j_k}$ by minimizing (6.9). Only if the two estimates are statistically similar, i.e., if

$$e(\boldsymbol{\xi}_{j_k i}, \boldsymbol{\xi}_{i j_k}) := (\boldsymbol{\xi}_{j_k i} \circ \boldsymbol{\xi}_{i j_k})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\xi}_{j_k i} \circ \boldsymbol{\xi}_{i j_k}) \tag{6.18}$$

$$\text{with} \quad \boldsymbol{\Sigma} := \boldsymbol{\Sigma}_{j_k i} + \text{Adj}_{j_k i} \boldsymbol{\Sigma}_{i j_k} \text{Adj}_{j_k i}^T \tag{6.19}$$

	SLAM				VO		
Seq.	t_{rel}	r_{rel}	t_{abs}	time	t_{rel}	r_{rel}	time
00	0.63	0.26	1.0	82	1.09	0.42	21
01	2.36	0.36	9.0	37	2.13	0.37	24
02	0.79	0.23	2.6	64	1.09	0.37	28
03	1.01	0.28	1.2	72	1.16	0.32	27
04	0.38	0.31	0.2	51	0.42	0.34	28
05	0.64	0.18	1.5	77	0.90	0.34	29
06	0.71	0.18	1.3	72	1.28	0.43	29
07	0.56	0.29	0.5	74	1.25	0.79	31
08	1.11	0.31	3.9	73	1.24	0.38	29
09	1.14	0.25	5.6	61	1.22	0.28	30
10	0.72	0.33	1.5	70	0.75	0.34	21
mean 00-10	0.91	0.27	2.6	67	1.14	0.40	29
mean 11-21	1.21	0.35	–	69	1.40	0.36	28

- t_{rel}: translational RMSE drift (%), av. over 100 m to 800 m intervals.
- r_{rel}: rotational RMSE drift (deg per 100 m), av. over 100 m to 800 m intervals.
- t_{abs}: absolute RMSE after 6DoF alignment, in meters.
- time: single-threaded computation time per frame, in milliseconds.

Table 6.1: **Results on Kitti Benchark.**

is sufficiently small, they are added as constraints to the pose-graph. Here, $\text{Adj}_{j_k i}$ is the adjoint of $\boldsymbol{\xi}_{j_k i}$ in SE(3). To speed up the removal of incorrect loop-closure candidates, we apply this consistency check after each pyramid level. Only if it passes, direct image alignment is continued on the next higher resolution. This allows to discard most incorrect candidates with only very little wasted computational resources: Figure 6.6 shows how many constraints where tracked on which pyramid level for one of the longest sequences in the Kitti dataset.

6.4 Results

We present the results obtained by Stereo LSD-SLAM (1) on the well-known Kitti dataset, and (2) on three sequences recorded from a micro aerial vehicle (MAV) flying indoors, taken from the EuRoC Challenge 3. We evaluate both the runtime and accuracy, for different parameter settings. Although our implementation makes heavy use of multiple CPU cores, all timings given in this chapter refer to single-threaded execution on an Intel i7-4900MQ CPU running at 2.8 Ghz.

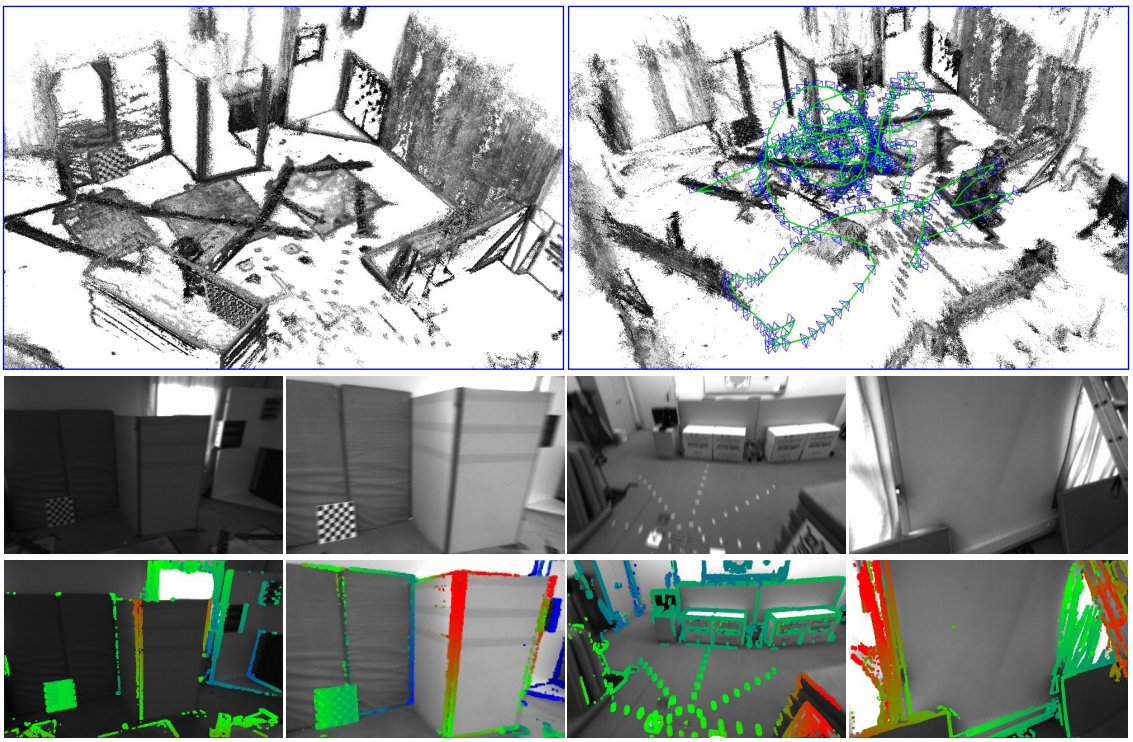


Figure 6.7: **Results on EuRoC Datasets.** Top: reconstruction from the first (left) and third (right) trajectory. Bottom: Selection of images from the third trajectory, displaying strong lightning changes (first to second image), motion blur (third image) and views with little texture (fourth image).

6.4.1 EuRoC Dataset

We run Stereo LSD-SLAM on the EuRoC dataset, taken from a MAV flying around a room which is equipped with a motion capture system for ground truth acquisition. The dataset contains 3 trajectories, with increasingly aggressive motion. Fig. 6.7 shows the reconstruction obtained. The absolute translational RMSE is 6.6 cm, 7.4 cm and 8.9 cm for the first, second and third trajectory respectively. In this dataset we removed the first and last 150 images for each trajectory, as in some of them only the ground surface is visible.

6.4.2 Kitti Dataset

We evaluated our method on the well-known Kitti dataset. Table 6.1 summarizes the results both for Stereo LSD-SLAM with, and without loop-closures (VO). The results given are for half resolution, as we feel this is a better trade-off between accuracy and computational speed – see also Sec. 6.4.4. On the evaluation sequences 11-21, we achieve a mean translational RMSE of 1.21% for full SLAM, which currently ranks second amongst stereo methods. Stereo LSD-SLAM is however much faster

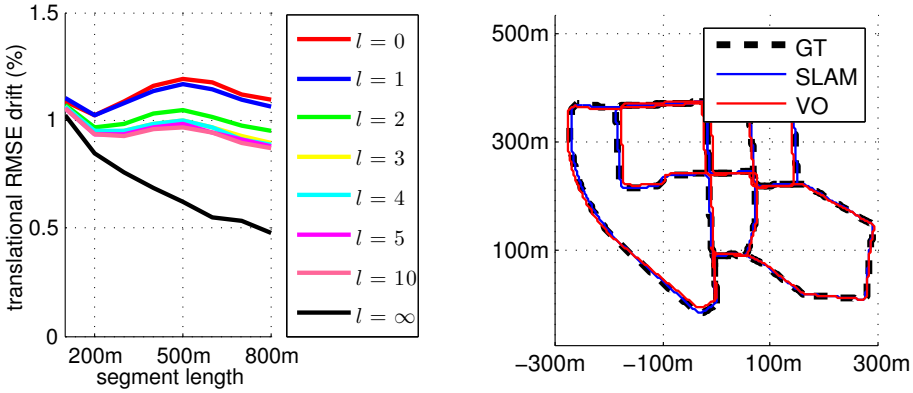


Figure 6.8: **Visual Odometry vs. SLAM.** Left: translational drift over different evaluation segment lengths, for different sizes of the pose-graph optimization window l. For $l = \infty$, our method performs full SLAM; hence the translational drift decreases when evaluating over longer segments (down to 0.5%). Right: 6DoF-aligned trajectories of the Kitti 00 sequence. While performing local pose-graph optimization slightly increases the local accuracy, it cannot remove drift over long segments.

than methods achieving similar accuracy. The increased error compared to the test sequences 00-10 is due to the presence of many moving objects in 20 and 21, which cause direct image alignment to occasionally fail (Sec. 6.4.6). Furthermore, the Kitti benchmark only provides images captured at 10 Hz while driving at speeds of up to 80 $^{\text{km}}/_{\text{h}}$ – which is challenging for direct methods, as these are good at exploiting small intra-frame motions.

6.4.3 Visual Odometry vs. SLAM

Here, we evaluate the capability to perform large-scale loop-closures when running the full SLAM system, as well as the effect of only performing loop-closures in a small window of the last l frames – effectively turning Stereo LSD-SLAM into a Visual Odometry. For $l = 0$, no image alignment with geometric error is performed, and only the pose from the initial frame alignment is used. For this comparison, we only consider Kitti sequences which contain significant loop-closures, i.e. 00, 02, 05, 06 and 07. Figure 6.8 summarizes the result: It can clearly be seen that performing full SLAM greatly decreases long-term drift, which is little surprising. However, this comes at increased computational cost: when performing full SLAM, the overall computational budget required more than doubles (also see Tab. 6.1), as the full pose-graph has to be optimized and many loop-closure constraints have to be tracked. All numbers in this Section refer to running Stereo LSD-SLAM at half resolution.

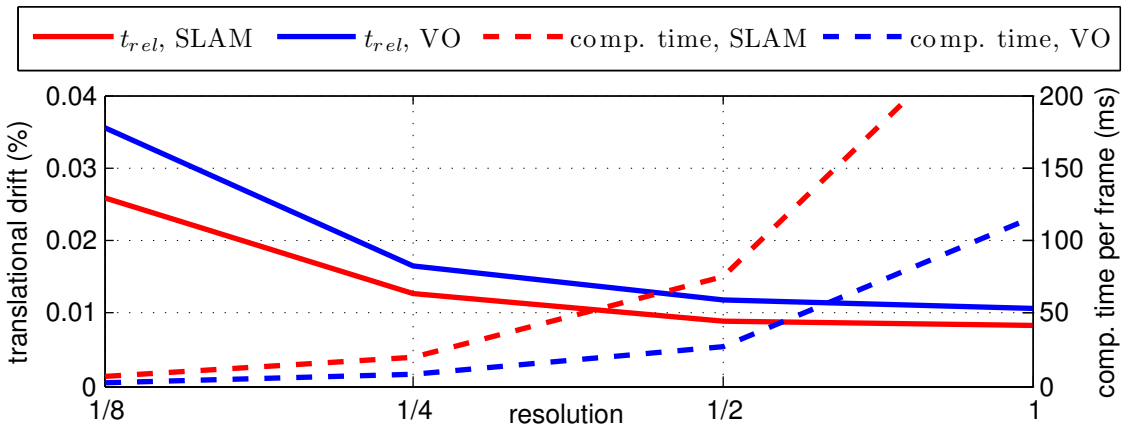


Figure 6.9: **Image Resolutions.** The plot shows the mean translational RMSE t_{rel} for different image resolutions, as well as the required computation time. Stereo LSD-SLAM allows to smoothly trade-off one for the other – for an image resolution of one eight of the original, it runs at 400 Hz (VO) / 145 Hz (SLAM) in a single thread, still achieving a mean drift of only 3.5% (VO) and 2.5% (SLAM).

6.4.4 Effect of Image Resolution

A beautiful property of Stereo LSD-SLAM is that the achieved accuracy degrades very gracefully with decreasing image resolution, while the computational budget required shrinks rapidly. In fact, we were able to run both full SLAM as well as VO on the Kitti dataset at down to one eight of the original resolution, i.e., 154×46 pixels, and still achieve a reasonable mean translational drift of 2.5% (SLAM) and 3.5% (VO) – at greatly reduced computational cost, running in 15× real-time (SLAM) and 40× real-time (VO). The result is summarized in Fig. 6.9.

6.4.5 Performance Analysis

In Table 6.2, we summarize the computational time required for each part of the algorithm. All timings are given in milliseconds per frame. For lower resolutions, images are down-sampled in a pre-processing step, as this typically can be done at no additional cost in hardware (pixel binning). It can clearly be observed that all parts of the algorithm – except for pose-graph optimization – directly scale with the number of pixels in the image. Only at very low resolution, resolution-independent operations – like inverting the Hessian during LM minimization – start to have a visual impact.

6.4.6 Moving Objects & Occlusions

A remarkable property of direct image alignment approaches is the ”locking property” [57]: In the presence of multiple motions or outliers, the coarse-to-fine approach

	154×46	310×92	620×184	1240×368
Tracking	1.2 ms	4.2 ms	16.0 ms	61.0 ms
Mapping	0.8 ms	2.9 ms	13.1 ms	62.8 ms
Constr. Search	3.7 ms	10.5 ms	40.0 ms	143.1 ms
Pose-Graph Opt.	1.2 ms	1.3 ms	1.4 ms	1.3 ms
Total (SLAM)	6.9 ms	18.9 ms	70.5 ms	268.2 ms

Table 6.2: **Computational Time Required.**

causes direct methods to lock onto the most dominant motion within the validity radius of the linearisation. A robust weighting function then allows to minimize the effect of pixels not belonging to this motion. Figure 6.10 shows three examples in which large parts of the image are moving or become occluded: In the first two examples the dominant motion is correctly identified, whereas in the third example image alignment locks onto the moving cars in the foreground. We observed this problem only in Sequence 20 of the Kitti benchmark as there are many cars moving at the same speed – arguably making the dominant motion in the scene that of the cars. For the on-line evaluation, we resolve this by removing all points in a certain volume in front of the car for this sequence only. Nevertheless, future work could take advantage of our approach, for example by segmenting the scene motion into a number of rigid-body motions ([57, 111, 124]).

6.4.7 Qualitative Results

We show in Fig. 6.11 some qualitative results of the estimated semi-dense depth maps, and the resulting point-clouds. Note how depth is estimated in almost all areas that have gradient information, and how many fine details (signs, lamp posts) are recovered. Also, the inclusion of temporal stereo allows to estimate depth for strictly horizontal structures, like the power transmission lines visible in some of the images.

6.5 Conclusion

We proposed Stereo LSD-SLAM, a novel direct approach to SLAM with stereo cameras. Our method leverages static, fixed-baseline stereo as well as temporal, variable-baseline stereo cues. Static stereo provides accurate depth within the effective operating range of the stereo camera. It also removes scale ambiguities and difficulties with degenerate motion along the line of sight, a problem inherent to monocular SLAM that only uses temporal stereo. With temporal stereo on the other hand, depth can be estimated in variable baseline directions that correspond to the translational motion between frames.

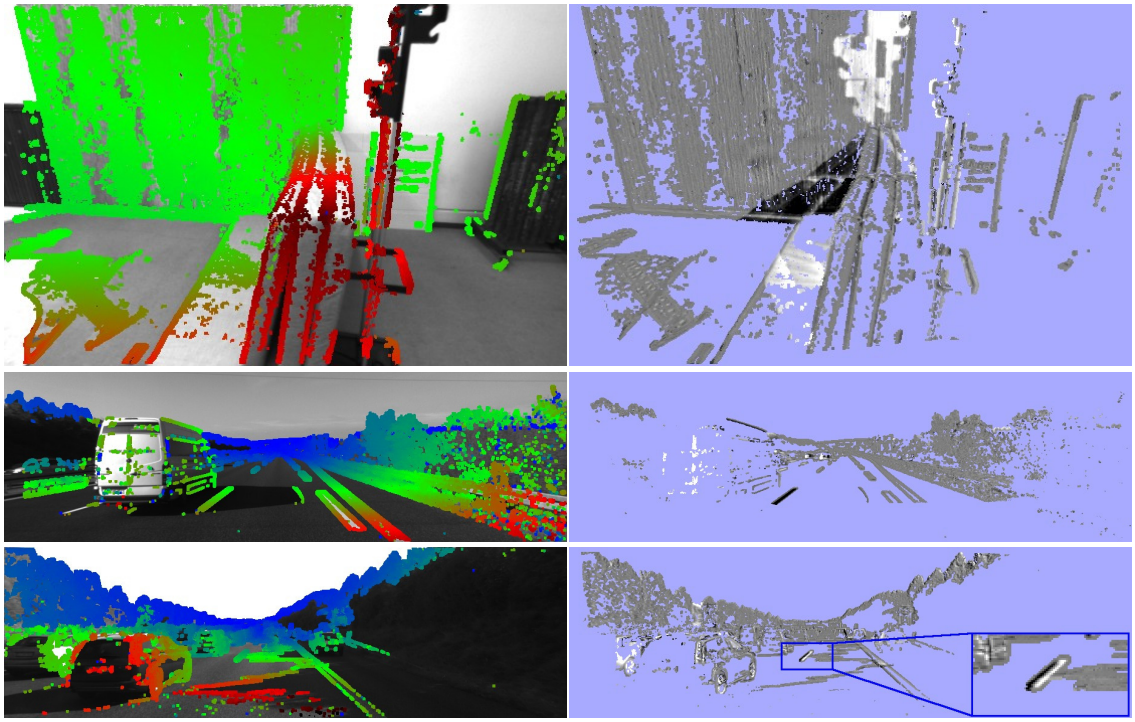


Figure 6.10: **Failure Cases.** Examples for scenes with moving objects & strong occlusions. On the right, we show the intensity residual after direct image alignment (small values are shown in gray; large negative / positive residuals are shown in black / white). While in the first two examples direct image alignment locks onto the correct motion, in the last one, it latches onto the wrong motion in the scene – the moving cars – and fails to align the two images correctly. This can be seen by the residual around the lane marking.

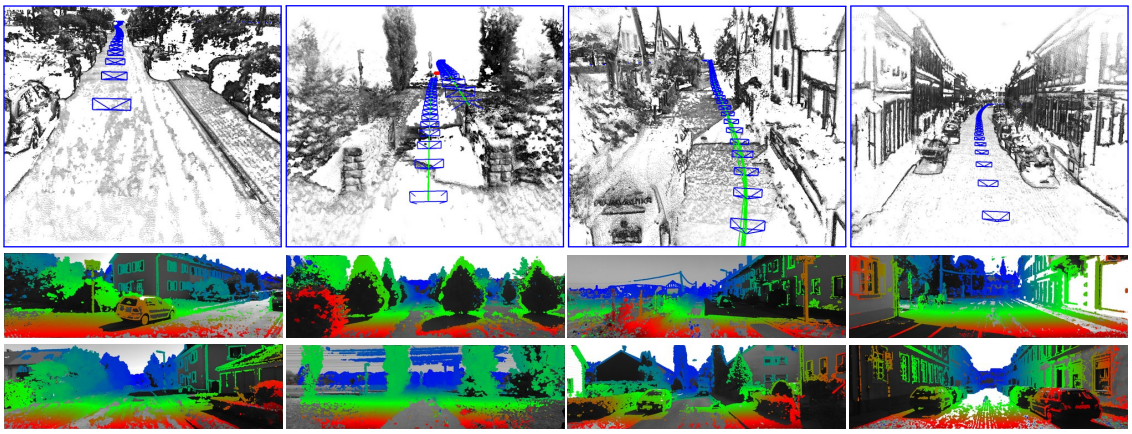


Figure 6.11: **Qualitative Examples.** Point clouds and depth maps for the Kitti dataset (sequences 08,14,15,18), running at full resolution. Also see the attached video.

Our method directly aligns images using photometric and geometric residuals at a semi-dense set of pixels. We choose pixels where there is sufficient information for static or temporal stereo estimation. In contrast to sparse interest-point-based methods, our approach is not restricted to a specific type of image features that are extracted in a decoupled processing stage prior to image alignment.

In our experiments, Stereo LSD-SLAM demonstrates state-of-the-art results on the popular Kitti benchmark dataset for stereo odometry and SLAM on autonomous cars. Stereo LSD-SLAM also performs very accurate on challenging sequences recorded with a micro aerial vehicle (MAV) for the EuRoC Challenge 3. Both datasets are very challenging for a purely monocular SLAM approach, since motion is mainly along the line of sight (cars), or can mainly consist of rotations (MAVs).

In future work, we consider extending our approach to multi-camera setups beyond binocular stereo cameras. Sensor fusion with inertial or GPS information could further enhance accuracy and robustness on the local and the global scale. Finally, we plan to address multi-body motion segmentation and estimation. This way, our method would not only recover the dominant motion in the images, but also the motion of further independent moving objects.

Chapter 7

Large-Scale Direct SLAM for Omnidirectional Cameras

Authors	David Caruso[2]	david.caruso@polytechnique.edu
	Jakob Engel[1]	engelj@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] Ecole Polytechnique Paris	

Publication **Large-Scale Direct SLAM for Omnidirectional Cameras**. D. Caruso, J. Engel, D. Cremers In *Preceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), 2015.*

doi: 10.1109/IROS.2015.7353366

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

Abstract. We propose a real-time, direct monocular SLAM method for omnidirectional or wide field-of-view fisheye cameras. Both tracking (direct image alignment) and mapping (pixel-wise distance filtering) are directly formulated for the unified omnidirectional model, which can model central imaging devices with a field of view well above 150°. This is in stark contrast to existing direct mono-SLAM approaches like DTAM or LSD-SLAM, which operate on rectified images, limiting the field of view to well below 180°. Not only does this allows to observe – and reconstruct – a larger portion of the surrounding environment, but it also makes the system more robust to degenerate (rotation-only) movement. The two main contribution are (1) the formulation of direct image alignment for the unified omnidirectional model, and (2) a fast yet accurate approach to incremental stereo directly on distorted images. We evaluated our framework on real-world sequences taken with a 185° fisheye lens, and compare it to a rectified and a piecewise rectified approach.

7.1 Introduction

Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) are becoming increasingly important for robotics and mobile vision applications, as they only require optical cameras – which are cheap, light and versatile, and hence can easily be put into commodity hardware. A lot of research has been focused around these topics throughout the last decade, with a particular focus on real-time systems – which can be used for autonomous control for example of UAVs [7], [14].

Most existing approaches are based on keypoints: Once keypoints are extracted, the images are abstracted to a collection of point-observations which are then used to compute geometrical information. This can be done in a filtering framework [31][78][23], or in a keyframe-based non-linear optimization framework [69], [74], [108]. This arguably has the advantage that a large part of the required workload only is done once on keypoint extraction, such that remaining computational resources can be spent on enforcing large-scale geometric consistency (Bundle Adjustment), and outliers can be removed in a straight-forward way.

More recently, so-called direct approaches have gained in popularity: instead of abstracting the images to point-observations, they compute dense [88], or semi-dense [4] depth maps in an incremental fashion, and track the camera using direct image alignment. This has the advantage that much more information can be used, in particular information contained edges or densely textured surfaces. Further, the generated map contains substantially more information about the environment, which can be used for obstacle-avoidance and path-planning.

What all these visual methods have in common, is that they rely on a sufficiently informative observed environment. In many practical cases however, this can be a

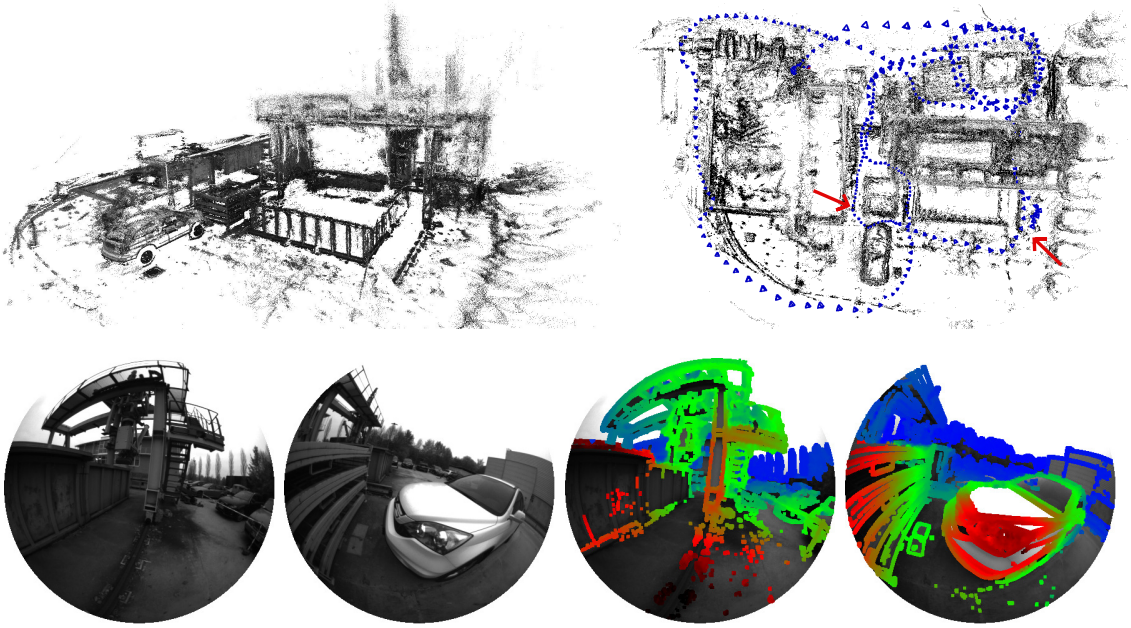


Figure 7.1: **Reconstruction from Omnidirectional Camera.** Top: 3D reconstruction obtained in real-time with our approach, using a 185° fisheye lens. Bottom: Map of the trajectory and set of example keyframes, with associated color-coded inverse distance maps.

very restrictive assumption: For example indoors where there are many untextured white walls, or in the presence of moving objects, large parts of the camera image can become uninformative for SLAM. This is especially true if the used camera only has a small field of view (FoV). On the other hand, the wider the field of view, the more likely that some part of the visible scene is well-suited for SLAM.

Nevertheless, most visual SLAM or VO systems are restrained to using a classical pinhole camera model. Often, this is combined with a radial distortion model (such as the ATAN model used in PTAM). All these models can not directly be used for omnidirectional camera (with FoV of more than 180°). This is especially true for direct methods, which typically operate on rectified images – limiting the field of view to no more than 130°.

In this paper, we propose an extension of LSD-SLAM [4] to a generic omnidirectional camera model. The resulting method is capable of handling all types of central projection systems such as fisheye and catadioptric cameras. We evaluate it on images captured with a fisheye lens covering a FoV of 185°. We show that especially for trajectories which contain aggressive camera rotations, it outperforms in the previously presented algorithms, without losing its real-time capability.

7.1.1 Related Work

There is a range of related work regarding omnidirectional vision, in particular for robot and ground-vehicle localization. For instance [103] uses a catadioptric system to estimates the ego-motion of a vehicle, using direct photometric error minimization for rotation estimation – it is however restricted to planar motion. In [116], RANSAC point association for SIFT features is used for estimating translation and rotation, on a rig of 5 rectified cameras. Again, the system is restricted to planar motion. In [105] a multicamera rig is used to build a topological map based on appearance. In [50] an EKF-based SLAM system is adapted for omnidirectional cameras; In [96], the advantage of using omnidirectional cameras in this context is shown. The work of Meilland et. al. [84] is somewhat closer to ours, as it performs dense registration against multiple frames from a database of spherical images. They are augmented with distance information from an external sensor or stereo-vision. However, the system is based on a priori learned database of georeferenced images and does not perform online SLAM.

7.1.2 Contribution and Outline

In this paper we explore the use of omnidirectional and fisheye cameras for direct, large-scale visual SLAM. We propose two different camera model choices, which we integrate into the recently appeared LSD-SLAM [4] framework, and evaluate the resulting algorithm on real-world and simulated data. More precisely, the main contribution of this paper is two-fold: (1) We give a direct image alignment formulation operating on an omnidirectional camera model. (2) We derive an efficient and accurate approach to perform stereo directly on omnidirectional images, both for the piecewise rectification approach and natively on the Unified Omnidirectional Model. We intend to make the used datasets including ground-truth publicly available.

The paper is organized as follows: In Chapter 7.2, we introduce a camera model as general projection function, and describe the three parametrized models considered in this paper: The *Pinhole Model* in Sec. 7.2.1, an *Array of Pinhole Models* in Sec. 7.2.2, and the *Unified Omnidirectional Model* in Sec. 7.2.3. In Chapter 7.3, we describe our omnidirectional direct SLAM method. We start by reviewing the LSD-SLAM pipe-line as introduced in [4]. We then detail how the two major steps that depend on the camera model – probabilistic, semi-dense depth estimation and direct image alignment – are adapted to operate in real-time on images from omnidirectional cameras. In Chapter 7.4, we evaluate the accuracy, robustness and runtime for the three different models on a both simulated and real-world data. Finally, in Chapter 7.5, we summarize the results and line out future work.

Figure 7.2: **Camera Models.** The same image, warped to fit the three projection models considered in this paper. While the Unified model and the piecewise rectified model can cover the full 185 ° field-of-view, the pinhole model shows significant distortion – when cropping the image to a field-of view of only 120 ° horizontally, as done for this figure.

7.2 Camera Models

In this chapter, we will lay out the three different parametric projection functions π considered in the paper: In Sec. 7.2.1, we briefly review the well-known *Pinhole Model* and discuss its limitations. We then extend it to a more general *Array of Pinhole Models* allowing to cover the full viewing sphere in Sec. 7.2.2. In Sec. 7.2.3, we introduce the Unified Omnidirectional Model, which allows to model 360 °-vision in closed-form.

Notation. We use bold, capital letters $\boldsymbol{R}$ to denote matrices, and bold, lower-case letters $\mathbf{x}$ for vectors. $\mathbf{u} = [u, v]^T \in \Omega \subset \mathbb{R}^2$ will generally denote pixel coordinates, where Ω denotes the image domain. $\mathbf{x} = [x, y, z]^T \in \mathbb{R}^3$ will be used for 3D point coordinates and $\tilde{\mathbf{x}} := [\mathbf{x}^T, 1]^T$ for the corresponding homogeneous point. $[\cdot]_i$ denotes the i'th row of a matrix / vector.

In the most general case, a camera model is a function $\pi \colon \mathbb{R}^3 \to \Omega$, which defines the mapping between 3D points $\mathbf{x}$ in the camera frame, and pixels $\mathbf{u}$ in the image. For lenses with negligible diameter, a common assumption is the *single viewpoint assumption*, i.e., that all light-rays pass through a single point in space – the origin of the camera frame $\mathcal{C}$. Hence, the projected position of the point only depends on the direction of $\mathbf{x}$. We will use $\pi^{-1} \colon \Omega \times \mathbb{R}^+ \to \mathbb{R}^3$ for the function mapping pixels back to 3D, using their inverse distance d. Further, we define a directed orthonormal camera frame centred at $\mathcal{C}$ by fixing a privileged direction $\mathbf{z}$ (the *principal axis*) and two other orthogonal directions.

Note that the single viewpoint assumption allows transforming images from any

camera model to any other, for the domain of visible points they have in common – this is generally referred to as *image rectification*, and is a frequently done preprocessing step, transforming the image to follow a more simple model e.g. by removing radial distortion. Given two projection functions π_1, π_2 and an image $I_1 \colon \Omega_1 \to \mathbb{R}$ taken with a camera π_1, we can compute the respective image $I_2 \colon \Omega_2 \to \mathbb{R}$ following projection π_2 as

$$I_2(u, v) = I_1(\pi_1(\pi_2^{-1}(u, v))) \tag{7.1}$$

This warping however introduces interpolation artifacts and can degrade the image quality, especially in areas where the angular resolution changes significantly.

7.2.1 Pinhole Model

The pinhole camera model is the most used camera model. The image is obtained by projecting each point onto a plane located at $z = 1$, followed by an affine mapping

$$\pi_p(\mathbf{x}) := \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} x/z \\ y/z \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{7.2}$$

where f_x, f_y are the focal lengths, and c_x, c_y is the principal point. It is schematically shown in Fig. 7.3.

This model is often used as the linearity of the projection function (in homogeneous coordinates) – and the fact that straight lines in 3D are projected to straight lines in the image – make it the most simple model choice to use. It however has the major drawback that it cannot model a wide field of view: The angular resolution decreases drastically towards the borders of the image, leading to a distorted image – an example is shown on the right in Fig. 7.7.

In order to make this model compatible to small radial distortions, a non-linear radial distortion function – often approximated polynomially – can be applied to the projected pixel coordinates. Still, the nature of a pinhole projection forbids points to lie behind the image plane, limiting the field of view to below 180 °.

7.2.2 Array of Pinhole Camera

A straight-forward approach to extending the field of view is to use a camera model consisting of an array of several pinhole cameras, which have the same principal point but different orientations. The projection function $\pi_{mp}(\mathbf{x}) \colon \mathbb{R}^3 \to \cup_i \Omega_i$ is then given by piecewise rotation followed by pinhole projection, i.e.,

$$\pi_{mp}(\mathbf{x}) := \pi_{p_{i(\mathbf{x})}}(\mathbf{R}_{i(\mathbf{x})}\mathbf{x}) \tag{7.3}$$

where $i(\mathbf{x}) \colon \mathbb{R}^3 \to [1, k]$ segments the 3D space into k subspaces. While in general the segmentation and orientation of the associated cameras can be chosen arbitrarily,

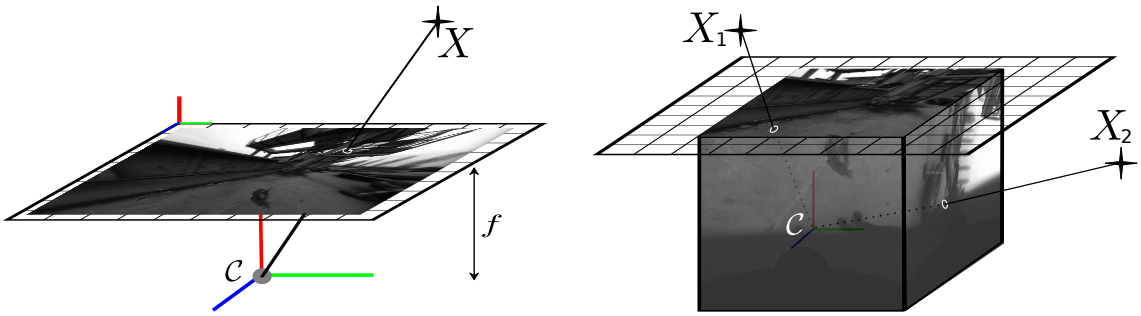


Figure 7.3: **Pinhole and Piecewise Pinhole Projection.** Left: Pinhole Model. A 3D point is directly projected onto the image plane through $\mathcal{C}$. Right: Piecewise Pinhole Model. A 3D point is projected through the center of the camera on one of the image planes depending on the subspace it lies in, effectively forming a cube-shaped image plane. X_1 and X_2 are projected to different images Ω_1 and Ω_2.

we choose to split $\mathbb{R}^3$ into six equally sized quadrants, forming a cube-shaped image plane. This has the advantage that $i(\mathbf{x})$ can be computed from binary comparisons on x, y and z, while the $\mathbf{R}_i$ correspond to orthogonal rotations.

While this model has a number of desirable properties – it is piecewise linear in homogeneous coordinates, simple to compute and offers reasonably homogeneous angular resolution – it does not fit natural lenses. In order to use it, incoming images have to be rectified in a preprocessing step. Further, the piecewise nature of the model causes discontinuities in the image space $\Omega = \cup_i \Omega_i$, complicating its use in practice.

7.2.3 Central Omnidirectionnal Camera: Unified Model

A number of different projection functions has been proposed in the literature for modeling and calibrating catadioptric and dioptric omnidirectional cameras. Desirable properties of such a function include (1) its capability to accurately describe a wide range of actual physical imaging devices, (2) the ease of parameter calibration and (3) the existence of a closed-form expression for the unprojection function π^{-1}. As this paper targets real-time direct SLAM, an additional criterion is the computational cost of projecting and unprojecting points, as well as the cost of evaluating the corresponding derivatives.

Accurate results were obtained by moving all non-linearities into a radially symmetric function, and identifying the first coefficients of its Taylor expansion [102]. While this approach can model every camera that fits the single viewpoint assumption, it lacks a closed-form unprojection function – and approximating it is computationally costly.

Instead, we use the model originally proposed in [46] for central catadioptric systems and extended in [123], [17] for a wider range of physical devices including

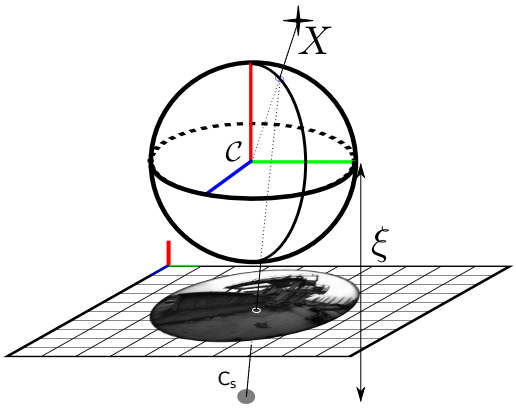


Figure 7.4: **Unified Model Projection Function.** A 3D point is first projected on the unit sphere, and then the image plane via a secondary, shifted camera center C_s.

fisheye camera. The central idea behind this model is to concatenate two successive projections: The first one projects the point from the world onto a camera-centered unit sphere. The second one is an ordinary pinhole projection trough a center shifted along the z axis by $-\xi$. This model is described by a total of five parameters, f_x, f_y, c_x, c_y and ξ. The projection of a point is computed as

$$\pi_u(\mathbf{x}) = \begin{bmatrix} f_x \dfrac{x}{z + \|\mathbf{x}\|\xi} \\ f_y \dfrac{y}{z + \|\mathbf{x}\|\xi} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{7.4}$$

where $\|\mathbf{x}\|$ is the euclidean norm of $\mathbf{x}$. The corresponding unprojection function can be computed in closed form, and is given by

$$\pi_u{}^{-1}(\mathbf{u}, d) = \frac{1}{d} \left(\frac{\xi + \sqrt{1 + (1 - \xi^2)\,(\tilde{u}^2 + \tilde{v}^2)}}{\tilde{u}^2 + \tilde{v}^2 + 1} \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix} \right), \tag{7.5}$$

where

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = \begin{bmatrix} (u - c_x)/f_x \\ (v - c_y)/f_y \end{bmatrix}. \tag{7.6}$$

One major advantage of this model is the availability of an easy-to-compute projection and unprojection function. Note that for $\xi = 0$ it reduces to the pinhole model. In order to improve the generality of the model, we we combine it with a small radial-tangential distortion to correct lens imperfections – similar to the pinhole case, images are warped once in the beginning, to perfectly fit this model.

7.3 Direct Omnidirectional SLAM

In this Chapter, we describe our omnidirectional, large-scale direct SLAM system, which is based on LSD-SLAM [4]. First, in Sec. 7.3.1 we review the LSD-SLAM pipeline adapted to omnidirectional cameras. We then derive a direct image registration formulation for the unified camera model in Sec. 7.3.2. In Sec. 7.3.4, we show how – in this framework – stereo can be done efficiently on the unified (1) and piecewise rectified (2) model.

Notation. $D\colon \Omega_d \to \mathbb{R}^+$ will denote the inverse distance map of the current keyframe.With a slight abuse of notation, elements of $\mathfrak{se}(3)$ will directly be represented as 6-vector $\boldsymbol{\mu}$, and we use exp and log to associate an element of the lie algebra to the corresponding element of the lie group. We then define the composition operator $\circ$ as

$$\boldsymbol{\mu_1} \circ \boldsymbol{\mu_2} := \log\left(\exp(\boldsymbol{\mu}_1) \cdot \exp(\boldsymbol{\mu}_2)\right). \tag{7.7}$$

As a shorthand, we use $\mathbf{R}_{\boldsymbol{\mu}}$ and $\mathbf{t}_{\boldsymbol{\mu}}$ to denote the corresponding rotation matrix and translation vector of a transformation, and $[\cdot]_i$ to extract the i'th row of a matrix/vector.

7.3.1 Method Overview

Our method continuously builds and maintains a pose-graph of keyframes. Each keyframe contains a probabilistic semi-dense inverse *distance* map, which maintains a Gaussian probability distribution over the inverse distance for all pixels which have sufficient intensity gradient. It is estimated over time by filtering over a large number of small-baseline stereo comparisons. In turn, new images – as well as loop-closure constraints – are computed using direct image alignment. Note that in contrast to [4], we use the inverse distance $d = \|\mathbf{x}\|^{-1}$ instead of depth, such that we can model points behind the camera. An overview is shown in Fig. 7.5.

SE(3) Tracking

When a new camera frame is captured, its rigid-body pose relative to the closest keyframe is tracked using direct image alignment, which will be described in 7.3.2.

Probabilistic Distance Map Estimation

Keyframes are selected at regular intervals, based on the moved distance to the previous keyframe (relative to its mean inverse distance), as well as the relative overlap. For each keyframe, an inverse distance map is initialized by propagating the inverse distance map from its immediate predecessor. Subsequently, it is updated

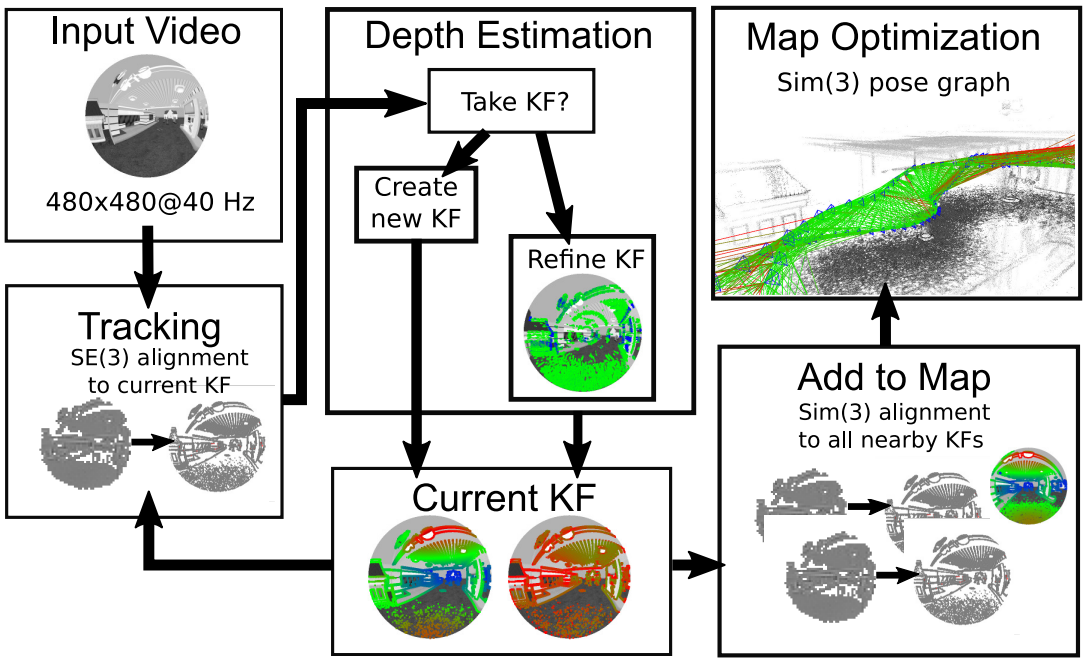


Figure 7.5: **Method Overview.** The LSD-SLAM pipeline for omnidirectional Cameras: Tracking and depth estimation inherently depend on the camera model used, their omnidirectional versions and are detailed in Sec. 7.3.2 and 7.3.4 receptively.

– and extended to new regions – by incorporating information obtained from many small-baseline stereo comparisons. This step will be described in more detail in Sec 7.3.4.

Scale-Drift Aware Pose-Graph Optimization

In the background, we continuously perform pose graph optimization between all keyframes, and attempt to find new constraints between keyframes which are likely to overlap. Constraints are expressed as similarity transforms to account for scale-drift – more details on this part can be found in [4].

Initialization

The system is initialized with a random depth map with mean one and a large covariance – this generally converges to a good estimate, as long as the camera motion within the first few seconds is not too degenerate.

7.3.2 Omnidirectional Direct Image Alignment on SE(3)

Every new frame I_{new} is tracked relative to the closest keyframe I_{Kf} with associated inverse distance map D_{Kf} by direct minimization of the photometric error, defined

as

$$E^{\text{frame}}(\boldsymbol{\mu}) := \sum_{\mathbf{u}\in\boldsymbol{\Omega}_\mathbf{d}} \rho\left(\left[\frac{r^I_\mathbf{u}(\boldsymbol{\mu})}{\sigma_{r^I_\mathbf{u}(\boldsymbol{\mu})}}\right]^2\right), \tag{7.8}$$

where ρ denotes the robust Huber norm, and with

$$r^I_\mathbf{u}(\boldsymbol{\mu}) = I_{\text{Kf}}(\mathbf{u}) - I_{\text{new}}\left(\pi(\omega(\boldsymbol{\mu},\mathbf{u}))\right) \tag{7.9}$$

$$\omega(\boldsymbol{\mu},\mathbf{u}) = \mathbf{R}_{\boldsymbol{\mu}}\pi^{-1}(\mathbf{u}, D_{\text{Kf}}(\mathbf{u})) + \mathbf{t}_{\boldsymbol{\mu}}. \tag{7.10}$$

The function ω unprojects a point, and transforms it by $\boldsymbol{\mu}$. As in [4], the residuals are normalized with their propagated inverse distance variance.

This weighted least-squares problem is then minimized in a coarse-to-fine scheme using the iteratively re-weighted Levenberg-Marquardt algorithm in a left-compositional formulation: In each iteration, we solve for a left-multiplied increment

$$\boldsymbol{\delta\mu}^{(k)} = \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\text{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})\right)^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r}, \tag{7.11}$$

where $\mathbf{r} = [\mathbf{r}^I_{\mathbf{u}_1}\dots\mathbf{r}^I_{\mathbf{u}_n}]^T$ is the stacked residual vector and $\mathbf{W}$ a diagonal matrix containing the weights. $\mathbf{J}$ is the $n\times 6$ Jacobian of the stacked residual vector evaluated at $\boldsymbol{\mu}^{(k)}$:

$$\mathbf{J} = \frac{\partial\mathbf{r}(\boldsymbol{\epsilon}\circ\boldsymbol{\mu}^{(k)})}{\partial\boldsymbol{\epsilon}} \tag{7.12}$$

which is then left-multiplied on the current estimate

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\delta\mu}^{(k)}\circ\boldsymbol{\mu}^{(k)}. \tag{7.13}$$

Using the chain rule, each 1×6 row $\mathbf{J}_\mathbf{u}$ of the Jacobian can be decomposed into three parts

$$\mathbf{J}^{\text{fwd}}_\mathbf{u} = -\mathbf{J}_{I_{\text{new}}}\Big|_\pi\mathbf{J}_\pi\Big|_\omega\mathbf{J}_\omega\Big|_{\boldsymbol{\mu}}, \tag{7.14}$$

where

- $\mathbf{J}_\omega\Big|_{\boldsymbol{\mu}^{(k)}}$ is a 3×6 Jacobian, denoting the left-compositional derivative of the transformed point, evaluated at $\boldsymbol{\mu} = \boldsymbol{\mu}^{(k)}$

$$\mathbf{J}_\omega\Big|_{\boldsymbol{\mu}} = \frac{\partial\omega(\boldsymbol{\epsilon}\circ\boldsymbol{\mu},\mathbf{u})}{\partial\boldsymbol{\epsilon}}. \tag{7.15}$$

- $\mathbf{J}_\pi\Big|_\omega$ is the 2×3 Jacobian of the projection function π evaluated at $\omega = \omega(\boldsymbol{\mu}^{(k)},\mathbf{u})$.

- $\mathbf{J}_{I_{\text{new}}}\Big|_{\pi}$ is the 1×2 intensity gradient of the new image, evaluated at point $\pi = \pi(\omega(\boldsymbol{\mu}^{(k)}, \mathbf{u}))$.

Notice how the evaluation point of each of these Jacobians depends on $\boldsymbol{\mu}^{(k)}$, hence everything has to be re-evaluated in each iteration. In practice, the computational cost is the dominated by this evaluation – which is especially true in our case, as for the unified model the projection, and hence its derivative $\mathbf{J}_{\pi}\Big|_{\boldsymbol{\omega}}$ is much more complex.

To avoid this, we use an *inverse compositional* formulation – a trick that is well known in the literature [16]: In each iteration, instead of applying the increment to the points in the reference frame, its inverse is applied to the points in the keyframe. That is, instead of linearizing

$$I_{\text{Kf}}(\mathbf{u}) - I_{\text{new}}(\pi(\omega(\boldsymbol{\epsilon} \circ \boldsymbol{\mu}^{(k)}, \mathbf{u}))), \tag{7.16}$$

with respect to $\boldsymbol{\epsilon}$, we linearize

$$I_{\text{Kf}}(\pi(\omega(\boldsymbol{\epsilon}, \mathbf{u}))) - I_{\text{new}}(\pi(\omega(\boldsymbol{\mu}^{(k)}, \mathbf{u}))). \tag{7.17}$$

The Jacobian now becomes

$$\mathbf{J}_{\mathbf{u}}^{\text{bkwd}} = \mathbf{J}_{I_{\text{Kf}}}\Big|_{\pi} \mathbf{J}_{\pi}\Big|_{\omega} \mathbf{J}_{\omega}\Big|_{\mathbf{0}}, \tag{7.18}$$

with $\omega = \omega(\mathbf{0}, \mathbf{u})$ and $\pi = \pi(\omega(\mathbf{0}, \mathbf{u}))$. It is thus independent of $\boldsymbol{\mu}^{(k)}$. This allows us to precompute it once per pyramid level, saving much of the required computations. Note that we still have to re-evaluate the outer product $\mathbf{J}^T\mathbf{W}\mathbf{J}$ on each iteration, as the weight matrix changes. The inverse of the resulting update is then right-multiplied onto the current estimate, i.e.,

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} \circ (-\boldsymbol{\delta}\boldsymbol{\mu}^{(k)}). \tag{7.19}$$

7.3.3 Omnidirectional Direct Image Alignment on Sim(3)

In monocular SLAM, the absolute scale is not observable and drifts over time – which has to be taken into account when finding loop-closures. As in [4], we use Sim(3) image alignment between keyframes, to estimate not only their relative pose, but also the scale difference between their inverse distance maps. This is done by introducing an additional error term – the geometric error – which penalizes differences in inverse distance. The energy function for aligning $(I_{\text{K1}}, D_{\text{K1}})$ and $(I_{\text{K2}}, D_{\text{K2}})$ thus becomes

$$E^{\text{Kf}}(\boldsymbol{\mu}) := \sum_{\mathbf{u} \in \boldsymbol{\Omega}_{\mathbf{d}}} \rho \left(\left[\frac{r_{\mathbf{u}}^{I}(\boldsymbol{\mu})}{\sigma_{r_{\mathbf{u}}^{I}(\boldsymbol{\mu})}} \right]^2 + \left[\frac{r_{\mathbf{u}}^{D}(\boldsymbol{\mu})}{\sigma_{r_{\mathbf{u}}^{D}(\boldsymbol{\mu})}} \right]^2 \right), \tag{7.20}$$

where $\boldsymbol{\mu} \in \mathfrak{sim}(3)$, and

$$r_{\mathbf{u}}^{I}(\boldsymbol{\mu}) = I_{\mathrm{K1}}(\mathbf{u}) - I_{\mathrm{K2}}\left(\pi(\omega_s(\boldsymbol{\mu}, \mathbf{u}))\right) \tag{7.21}$$

$$r_{\mathbf{u}}^{D}(\boldsymbol{\mu}) = \|\omega_s(\boldsymbol{\mu}, \mathbf{u})\|^{-1} - D_{\mathrm{K2}}\left(\pi(\omega_s(\boldsymbol{\mu}.\mathbf{u}))\right). \tag{7.22}$$

Note that we now optimize over relative scale as well, and hence have to apply a similarity warp, defined as

$$\omega_s(\boldsymbol{\mu}, \mathbf{u}) = s_{\boldsymbol{\mu}} \mathbf{R}_{\boldsymbol{\mu}} \pi^{-1}(\mathbf{u}, D_{\mathrm{K2}}(\mathbf{u})) + \mathbf{t}_{\boldsymbol{\mu}}, \tag{7.23}$$

where $s_{\boldsymbol{\mu}}$ is the scaling factor of $\boldsymbol{\mu}$. Note that in contrast to [4], this residual now penalizes differences in inverse distance. Again, we apply statistical normalization based on the propagated variances as in [4]. For tracking Sim(3)-constraints, we use a forward-compositional formulation.

We further note that as in [4], the approximated Hessian $(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}$ of the last iteration can be interpreted as covariance on a left-multiplied increment on $\boldsymbol{\mu}$, and is used in the subsequent pose-graph optimization.

7.3.4 Semi-Dense Depth Map Estimation

Once a frame is registered to a keyframe, stereo matching is performed to refine the keyframe distance map D_{Kf}. As matching cost we use the *sum of squared differences* (SSD) over five equidistant pixels along the epipolar line. If a prior exists, the epipolar search is constrained to the interval $[d - 2\sigma_d, d + 2\sigma_d]$. This greatly improves efficiency and minimizes the probability of finding an incorrect match, as in practice only very short line segments have to be searched. Subsequently, we refine the found match to sub-pixel precision.

Similar to [9], each new measurement is fused into the existing depth map. Measurement variances σ_m^2 are obtained using the geometric and photometric error, as derived in [9]. Finally, we smooth the inverse distance map, and remove outliers.

Non-Rectified Stereo

When performing stereo on the unified model, epipolar lines are not in fact lines but curves. More precisely, Geyer et al. showed that these epipolar curves are conics [46], as they are the pinhole-projection of a geodesic on the unit sphere, as visualized in Fig. 7.6. We here present a general method to incrementally and efficiently compute points along the epipolar curve, at a constant step-size of 1 px: While this is trivial for straight lines, it is not straight-forward for the general case of epipolar curves.

We first define the two points $\mathbf{p}_0, \mathbf{p}_\infty \in \mathbb{R}^3$ on the unit sphere around the projective center $\mathcal{C}_{\mathrm{ref}}$, which correspond to the maximum and minimum inverse distance of the search interval $d_{\max}$,$d_{\min}$:

$$\mathbf{p}_0 := \pi_s(\mathbf{R}\pi_u^{-1}(\mathbf{u}, d_{\max}) + \mathbf{t}) \tag{7.24}$$

$$\mathbf{p}_\infty := \pi_s(\mathbf{R}\pi_u^{-1}(\mathbf{u}, d_{\min}) + \mathbf{t}). \tag{7.25}$$

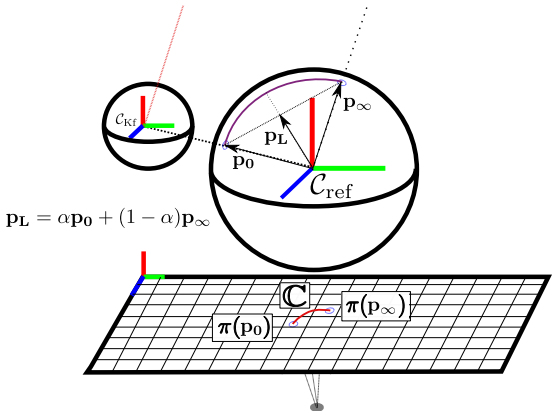


Figure 7.6: **Non-Rectified Stereo Matching.** We efficiently browse the epipolar curve in the image $\mathbf{u}_L$ using a parametric equation. It is obtained by projecting the line connecting $\mathbf{p}_0$ and $\mathbf{p}_\infty$ on the unit sphere around the camera center.

Here, π_s projects a point onto the unit sphere, π_u^{-1} is the unprojection function of the unified model (7.5), and $\mathbf{u}$ is the pixel in I_{Kf} we are trying to match. We then express the straight line between these points as

$$\mathbf{p}_L(\alpha) = \alpha\mathbf{p}_0 + (1-\alpha)\mathbf{p}_\infty, \tag{7.26}$$

for $\alpha \in [0,1]$. This also gives a parametric expression for the epipolar curve in I_{ref} as

$$\mathbf{u}_L(\alpha) := \pi_u(\mathbf{p}_L(\alpha)). \tag{7.27}$$

Note that we apply the full unified projection function, which first projects a point on $\mathbf{p}_L$ onto the geodesic, and then into the image. This is visualized in Fig. 7.6.

Starting at $\mathbf{u}_L(0)$, we then browse the epipolar curve by incrementing α. A step-size of 1 pixel is enforced by using a first-order Taylor expansion of $\mathbf{u}_L$, and choosing the increment in α as

$$\delta\alpha = \left\| \mathbf{J}_{\mathbf{u}_L}\Big|_\alpha \right\|^{-1}, \tag{7.28}$$

which we re-evaluate for each increment. Note that this method is independent of the shape of the epipolar curve, and hence can be used for any central camera model. Nevertheless, it is much more expensive than browsing a straight line, as each point is projected individually. In LSD-SLAM however, the search interval is always small, as either a good prior is available, or the pixel has just been initialized and hence the baseline is small.

Pre-Rectified Stereo

For a large disparity search range, the above method can become very costly since it requires re-evaluation of the projection function for each point. Thus, the valid

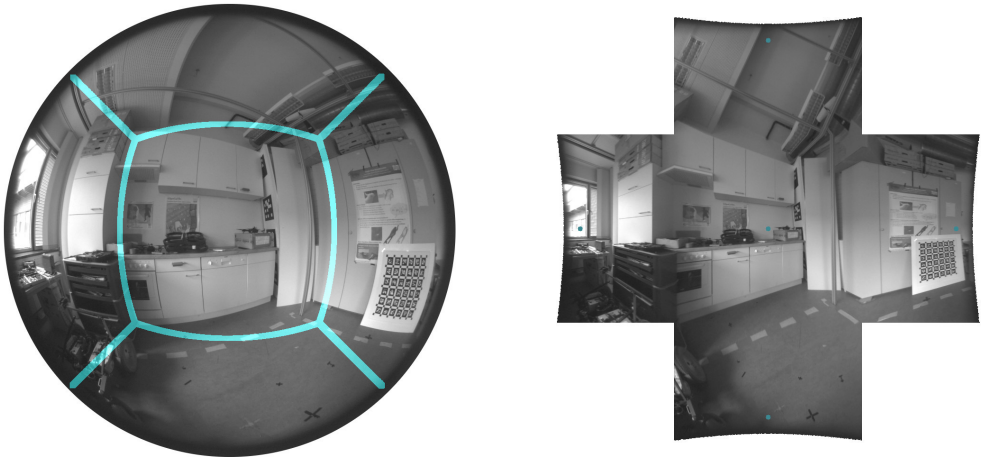


Figure 7.7: **Piecewise rectification.** Example of fisheye camera rectification. The borders are still distorted, as it is clearly visible on the checker-board, which leads to interpolation artifacts or blur.

question arises whether piecewise rectification of the input image as described in Sec.7.2.2, followed by straight-forward line-browsing would be faster. For this we determined suitable values for the focal lengths f_x and f_y of each pinhole camera individually, minimizing the change in angular resolution at each point in the image. An example is shown in Fig. 7.7: Still, some distortion is clearly visible, note for example how the checker-board shape is altered. Further, we extend the rectified images by extending their visible field by 20 pixels, which is not displayed in the figure. We then perform line-stereo the same way as is done in [9]. In Sec. 7.4 we will compare these two approaches regarding accuracy and efficiency.

7.4 Results

In this Chapter, we evaluate our algorithm regarding accuracy and computational requirements on both synthetic and real data. We first describe the experimental setup in. 7.4.1 and 7.4.2. We then evaluate the accuracy and the computational requirements in Sec. 7.4.3 and 7.4.4 respectively.

7.4.1 Hardware Setup

For real data experiment, we use a global shutter usb3 camera equipped with a 185 FoV fisheye lens. The ξ parameter for this system has been estimated to 2.06 by off-line calibration, using the *Kalibr* toolbox [82]. Images are cropped and scaled to a 480×480 region centered around the principal point. We recorded a number of trajectories with rapid, handheld motion, including quick rotation - an overview over one of the sequences can be seen Fig. 7.8. We also show two of the sequences (T2 and

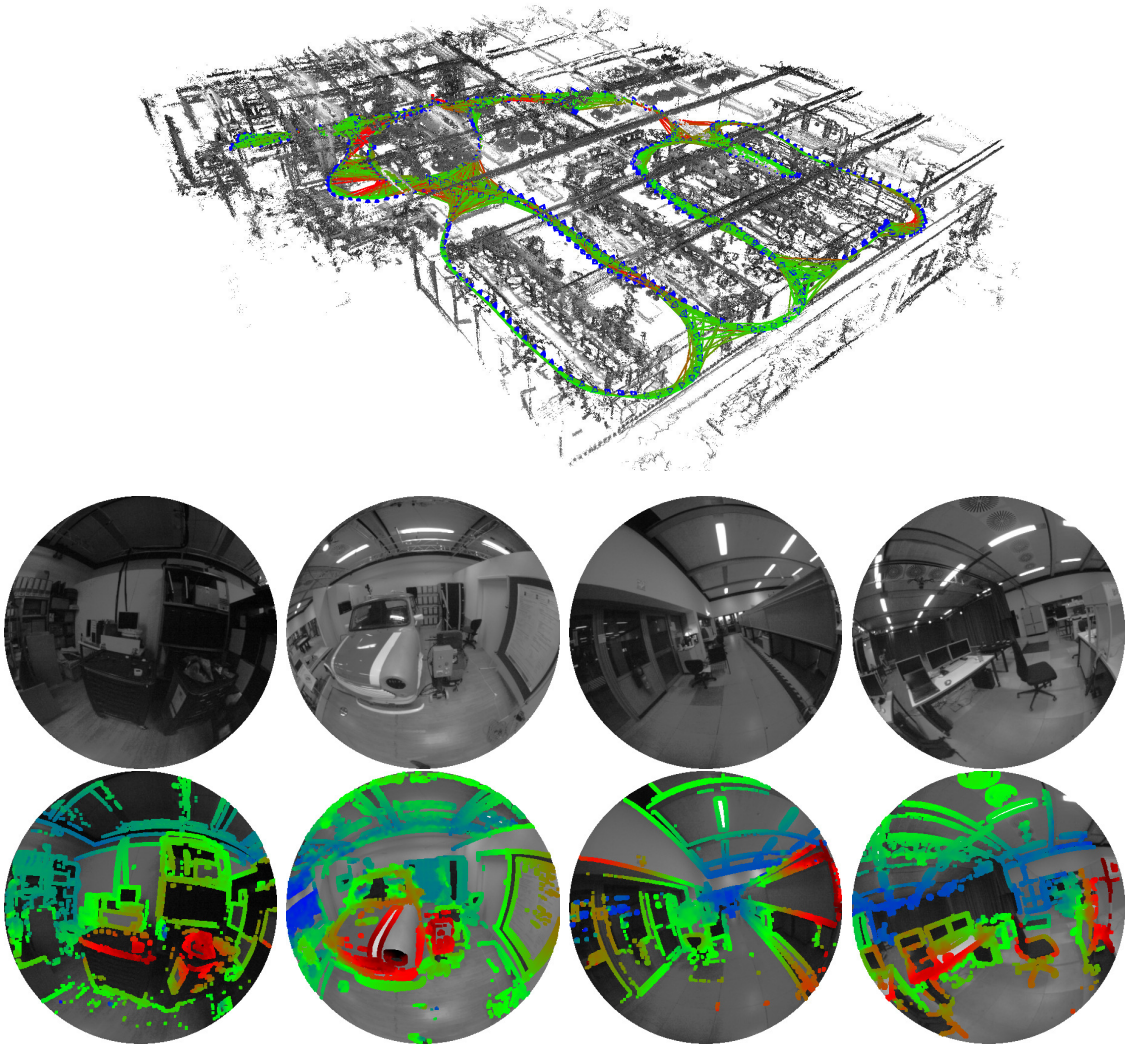


Figure 7.8: **Reconstruction of T5 sequence.** Top: Final point cloud. Bottom: Color-coded inverse distance maps. Note how we can obtain geometry for the full 185 ° field of view. This corresponds to the right plot in Fig. 7.9.

T5) in the attached video. For ground truth acquisition, we use a motion capture system which covers an area of approximately 7×12 m – as some trajectories leave this area, we only compute errors on the part for which ground truth data is available. The synthetic data was generated using the ROS gazebo simulator, modified to have as extra output the synchronized pose, 185° images, and distance ground truth. The movement is slower on this dataset and mimics that of a quadrotor. For comparison with a pinhole model, we also synthesize a sequence of rectified images, artificially cropping the field of view to 100° horizontally and vertically. We intend to make the dataset including ground truth publicly available.

7.4.2 Evaluated Parameters

We evaluate the effect of three different parameters:

- *Camera Model:* We use either the unified omnidirectional model (*Uni*, Sec. 7.3.4), or a piecewise rectified model (*Multi*, Sec. 7.3.4). As baseline, we use the cropped & rectified video with a pinhole model (*Pin*).
- *Input Resolution:* We use either an input resolution of 480×480 (*Full*) or 240×240 (*Half*).
- *Resolution Used for Tracking:* We choose to stop the coarse to fine approach in tracking either at the input image (level 0 of the image pyramid) or at the first octave of it (level 1), allowing to speed-up tracking significantly, while maintaining most of the accuracy.

All the experiments were conducted on a Intel i7 CPU, on a commercially available laptop.

7.4.3 Accuracy Comparison

In order to assess the accuracy of our method, we measure the translational root mean square error (RMSE) of the final position of all keyframes, after 7DoF alignment with the ground-truth. Because of the hard real-time constraint and the high frame-rate of the camera, frames may be dropped and different frames will be selected as keyframes – potentially having a significant effect on the result. We therefore average the RMSE of five runs. The results are shown in Tab. 7.2 and some representative visual result are shown in Fig. 7.9; also see the attached video.

Two things can be observed: First, results obtained with the omnidirectional camera clearly outperform the pinhole model. This shows that our algorithm can benefit from additional information in the image due to an increased field of view – the in some cases very significant difference is not surprising, as the recorded trajectories contain large amounts of rotation, which is very challenging for a normal camera.

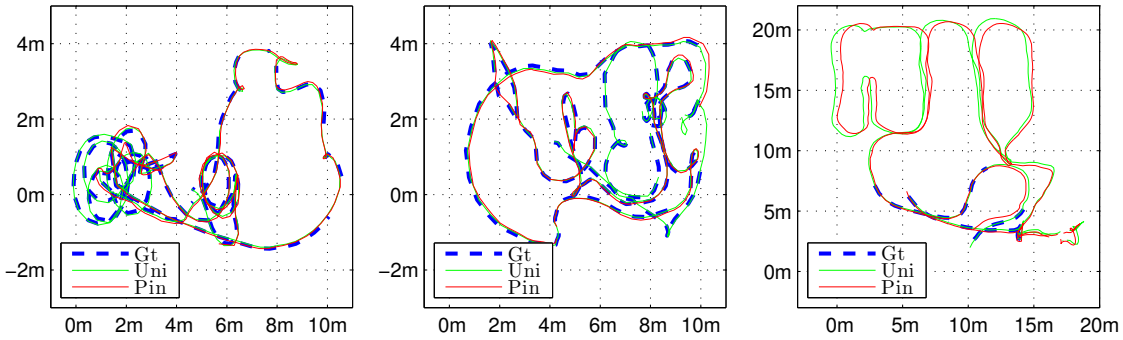


Figure 7.9: **Horizontal position for T2, T3 and T5.** The red line shows the result of Uni-Full-0, the green line that of Pin-Full-0, and the blue dotted line the ground truth where available. For T2 and T3, the pinhole version is lost for a large portion of the trajectory, as they include fast rotations which cannot be tracked well with the cropped field of view. For T5 the trajectory is correctly reconstructed with both camera models. The accuracy however is significantly better for the unified model (see Tab.7.2; The final pointcloud is shown in Fig. 7.8.

	480×480			240×240			160×160		
	Mul	Uni	Pin	Mul	Uni	Pin	Mul	Uni	Pin
Mapping	31	28	20	11	8	7	-	-	-
Tracking	24	24	17	10	10	6	3	3	2.2

Table 7.1: **Mean timing results in Milliseconds.**

The other observation is also expected: A higher resolution gives consistently better results than a lower resolution, although not by much. Interestingly, both half resolution omnidirectional methods outperform the full resolution pinhole model, showing that, at least in challenging scenes, a larger field of view is more important than high resolution. An example of 3D reconstruction with *half* and *full* resolution is given for the synthetic scene in Fig. 7.10.

7.4.4 Timing Measurement

Table 7.1 shows the measured average time taken by tracking and mapping. These times are measured on the same dataset as used for the accuracy assessment, and are in millisecond. These results shows that our distorted stereo matching algorithm is slightly more efficient than the multi rectified version. This is due to the rectification required beforehand, and the fact that almost always, the browsed epipolar segments do not exceed a couple of pixels in length. Real time is easily achieved since each frame can be tracked at least 40 Hz, and mapped at more than 30 Hz for a 480×480 image.

	T1	T2	T3	T4	T5	S1	S2
Mul-Full-0	**0.0493**	**0.0656**	**0.0456**	**0.0424**	**0.0535**	**0.0208**	**0.0484**
Mul-Full-1	0.0491	0.0699	0.0475	0.0479	0.0600	0.0387	0.0895
Mul-Half-0	0.0765	0.0966	0.0554	0.0546	0.0849	**0.0209**	**0.0433**
Mul-Half-1	0.0756	0.0977	0.0650	0.0952	0.1211	0.0345	0.1144
Uni-Full-0	0.0531	**0.0506**	**0.0463**	**0.0454**	**0.0358**	0.0340	0.0492
Uni-Full-1	**0.0508**	0.0634	0.0497	0.0514	0.0544	0.0429	0.0728
Uni-Half-0	0.0845	0.0731	0.0569	0.0588	0.0684	0.0382	0.0602
Uni-Half-1	0.1856	0.0837	0.0598	0.0730	0.1236	0.0428	0.0709
Pin-Full-0	0.5784	0.2282	0.0832	0.6049	0.5498	0.0474	0.7016
Pin-Full-1	0.6445	0.1526	0.0724	1.9756	0.9423	0.0861	0.6749
Pin-Half-0	1.1729	0.7301	0.6022	0.0863	1.1555	0.1297	1.5106
Pin-Half-1	1.5125	0.8351	0.6005	0.0941	2.3820	0.1685	1.1376

Table 7.2: **Absolute RMSE in Meters.**

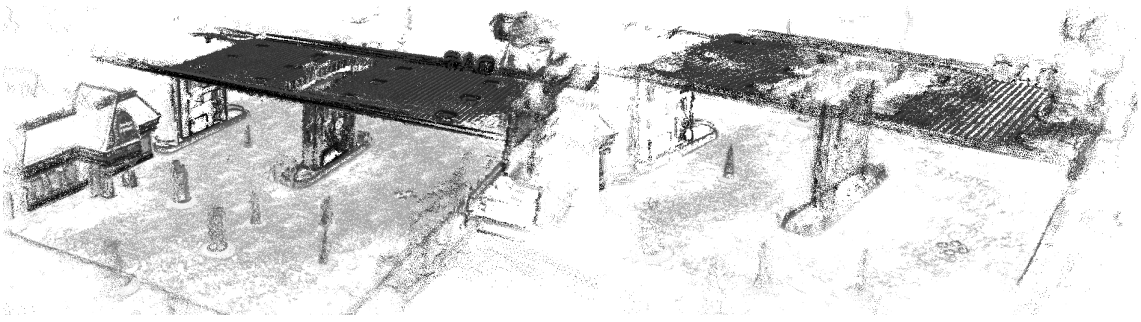


Figure 7.10: **Reconstruction evaluation.** Final point cloud obtained on S1 trajectory for two different resolutions (left 480×480, right 240×240): The resolution has a very sensitive impact on the completeness and accuracy of the 3D reconstruction.

7.5 Conclusion

We proposed in this paper a direct, semi-dense monocular SLAM system for omnidirectional cameras. Based on two different omnidirectional camera models, our system allows to directly use a wide range of classical dioptric or catadioptric imaging systems. The contribution of this paper is two-fold: (1) we explicitly formulate a camera model independent registration algorithm and (2) derived a generic, accurate, and efficient way to perform stereo, based on a parametric equation of the epipolar curves. We integrated these ideas into the LSD-SLAM framework and ran the algorithm in real-time on a number of videos captured by a 185° fisheye camera. We measure both an improvement of the accuracy of the localization and of its robustness to strong rotational movement compared to a standard camera. We also observe that even at relatively low resolutions (240×240), the localization accuracy surpasses the accuracy obtained when using a pinhole model, with a cropped field of view.

Chapter 8

Direct Sparse Odometry

Authors	Jakob Engel[1]	engelj@in.tum.de
	Vladlen Koltun[2]	vkoltun@gmail.com
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	
	[2] Intel Labs	
Publication	**Direct Sparse Odometry**. J. Engel, V. Koltun, D. Cremers Published as *arXiv:1607.02565 preprint*, July 2016.	
Contribution	Problem definition	*significantly contributed*
	Literature survey	*significantly contributed*
	Method development & evaluation	*significantly contributed*
	Implementation	*significantly contributed*
	Experimental evaluation	*significantly contributed*
	Preparation of the manuscript	*significantly contributed*

Abstract. We propose a novel direct sparse visual odometry formulation. It combines a fully direct probabilistic model (minimizing a photometric error) with consistent, joint optimization of all model parameters, including geometry – represented as inverse depth in a reference frame – and camera motion. This is achieved in real time by omitting the smoothness prior used in other direct methods and instead sampling pixels evenly throughout the images. Since our method does not depend on keypoint detectors or descriptors, it can naturally sample pixels from across all image regions that have intensity gradient, including edges or smooth intensity variations on mostly white walls. The proposed model integrates a full photometric calibration, accounting for exposure time, lens vignetting, and non-linear response functions. We thoroughly evaluate our method on three different datasets comprising several hours of video. The experiments show that the presented approach significantly outperforms state-of-the-art direct and indirect methods in a variety of real-world settings, both in terms of tracking accuracy and robustness.

8.1 Introduction

Simultaneous localization and mapping (SLAM) and visual odometry (VO) are fundamental building blocks for many emerging technologies – from autonomous cars and UAVs to virtual and augmented reality. Realtime methods for SLAM and VO have made significant progress in recent years. While for a long time the field was dominated by feature-based (indirect) methods, in recent years a number of different approaches have gained in popularity, namely *direct* and *dense* formulations.

Direct vs. Indirect. Underlying all formulations is a probabilistic model that takes noisy measurements $\mathbf{Y}$ as input and computes an estimator $\mathbf{X}$ for the unknown, hidden model parameters (3D world model & camera motion). Typically a Maximum Likelihood approach is used, which finds the model parameters that maximize the probability of obtaining the actual measurements, i.e., $\mathbf{X}^* := \operatorname{argmax}_{\mathbf{X}} P(\mathbf{Y}|\mathbf{X})$.

Indirect methods then proceed in two steps: First, the raw sensor measurements are pre-processed to generate an intermediate representation, solving part of the overall problem, such as establishing correspondences. Second, the computed intermediate values are interpreted as noisy measurements $\mathbf{Y}$ in a probabilistic model to estimate geometry and camera motion. Note that the first step is typically approached by extracting and matching a sparse set of keypoints – however other options exist, like establishing correspondencees in the form of dense, regularized optical flow.

Direct methods skip the pre-processing step and directly use the actual sensor values – light received from a certain direction over a certain time period – as

Figure 8.1: **Direct sparse odometry (DSO).** 3D reconstruction and tracked trajectory for a 1:40min video cycling around a building (monocular visual odometry only). The bottom-left inset shows a close-up of the start and end point, visualizing the drift accumulated over the course of the trajectory. The right images show some selected video frames.

measurements $\mathbf{Y}$ in a probabilistic model.

In the case of passive vision, the direct approach thus optimizes a *photometric error*, since the sensor provides photometric measurements. Indirect methods on the other hand optimize a *geometric error*, since the pre-computed values – point-positions or flow-vecors – are geometric quantities. Note that for other sensor modalities like depth cameras or laser scanners (which directly measure geometric quantities) direct formulations may also optimize a geometric error.

Dense vs. Sparse. Sparse methods use and reconstruct only a selected set of independent points (traditionally corners), whereas dense methods attempt to use and reconstruct all pixels in the 2D image domain. Intermediate approaches (semi-dense) refrain from reconstructing the complete surface, but still aim at using and reconstructing a (largely connected & well-constrained) subset of it.

Apart from the extent of the used image region however, a more fundamental – and consequential – difference lies in the addition of a geometry prior. In the sparse formulation, there is no notion of neighborhood, and geometry parameters (keypoint positions) are conditionally independent given the camera poses & intrinsics[1]. Dense (or semi-dense) approaches on the other hand exploit the conectedness of the used image region to formulate a geometry prior, typically favouring smoothness. In fact, such a prior is necessarily required to make a dense world model observable from passive vision alone. In general, this prior is formulated directly in the form of an additional log-likelihood energy term [88, 93, 112].

[1]Note that even though early filtering-based methods [31] kept track of point-point-correlations, these originated from marginalized camera poses, not from the model itself.

Note that the distinction between *dense and sparse* is not synonymous to *direct and indirect* – in fact, all four combinations exist:

- **Sparse + Indirect:** This is the most widely-used formulation, estimating 3D geometry from a set of keypoint-matches, thereby using a geometric error without a geometry prior. Examples include monoSLAM [31], PTAM [69], and ORB-SLAM [86].
- **Dense + Indirect:** This formulation estimates 3D geometry from – or in conjunction with – a dense, regularized optical flow field, thereby combining a geometric error (deviation from the flow field) with a geometry prior (smoothness of the flow field). Examples include [95, 119].
- **Dense + Direct:** This formulation employs a photometric error as well as a geometric prior to estimate dense or semi-dense geometry. Examples include DTAM [88], its precursor [112], and LSD-SLAM [4].
- **Sparse + Direct:** This is the formulation proposed in this paper. It optimizes a photometric error defined directly on the images, without incorporating a geometric prior. The motivation for exploring this combination is laid out in the following section.

8.1.1 Motivation

The **direct and sparse** formulation for monocular visual odometry proposed in this paper is motivated by the following considerations.

(1) Direct: One of the main benefits of keypoints is their ability to provide robustness to photometric and geometric distortions present in images taken with off-the-shelf commodity cameras. Examples are automatic exposure changes, non-linear response functions (gamma correction / white-balancing), lens attenuation (vignetting), de-bayering artefacts, or even strong geometric distortions caused by a rolling shutter.

At the same time, for all use-cases mentioned in the introduction, millions of devices will be (and already are) equipped with cameras solely meant to provide data for computer vision algorithms, instead of capturing images for human consumption. These cameras should and will be designed to provide a complete sensor model, and to capture data in a way that best serves the processing algorithms: Auto-exposure and gamma correction for instance are not unknown noise sources, but features that provide better image data – and that can be incorporated into the model, making the obtained data more informative. Since the direct approach models the full image formation process down to pixel intensities, it greatly benefits from a more precise sensor model.

One of the main benefits of a direct formulation is that it does not require a point to be recognizable by itself, thereby allowing for a more finely grained geometry

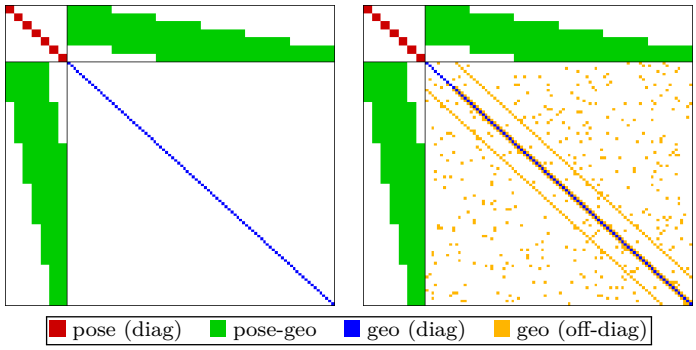


Figure 8.2: **Sparse vs. dense Hessian structure.** Left: Hessian structure of sparse bundle adjustment: since the geometry-geometry block is diagonal, it can be solved efficiently using the Schur complement. Right: A geometry prior adds (partially unstructured) geometry-geometry correlations – the resulting system is hence not only much larger, but also becomes much harder to solve. For simplicity, we do not show the global camera intrinsic parameters.

representation (pixelwise inverse depth). Furthermore, we can sample from across all available data – including edges and weak intensity variations – generating a more complete model and lending more robustness in sparsely textured environments.

(2) Sparse: The main drawback of a adding geometry prior is the introduction of correlations between geometry parameters, which render a statistically consistent, joint optimization in real time infeasible (see Figure 8.2). This is why existing dense or semi-dense approaches (a) neglect or coarsely approximate correlations between geometry parameters (orange), and / or between geometry parameters and camera poses (green), and (b) employ different optimization methods for the dense geometry part, such as a primal-dual formulation [88, 93, 112].

In addition, the expressive complexity of today's priors is limited: While they make the 3D reconstruction denser, locally more accurate and more visually appealing, we found that priors can introduce a bias, and thereby reduce rather than increase long-term, large-scale accuracy. Note that in time this may well change with the introduction of more realistic, unbiased priors learnt from real-world data.

8.1.2 Contribution and Outline

In this paper we propose a sparse and direct approach to monocular visual odometry. To our knowledge, it is the only fully direct method that jointly optimizes the full likelihood for all involved model parameters, including camera poses, camera intrinsics, and geometry parameters (inverse depth values). This is in contrast to hybrid approaches such as SVO [41], which revert to an indirect formulation for joint model optimization.

Optimization is performed in a sliding window, where old camera poses as well as points that leave the field of view of the camera are marginalized, in a manner inspired by [75]. In contrast to existing approaches, our method further takes full advantage of photometric camera calibration, including lens attenuation, gamma correction, and known exposure times. This integrated photometric calibration further increases accuracy and robustness.

Our CPU-based implementation runs in real time on a laptop computer. We show in extensive evaluations on three different datasets comprising several hours of video that it outperforms other state-of-the-art approaches (direct and indirect), both in terms of robustness and accuracy. With reduced settings (less points & active keyframes), it even runs at 5× real-time speed while still outperforming state-of-the-art indirect methods. On high, non-real-time settings in turn (more points & active keyframes), it creates semi-dense models similar in density to those of LSD-SLAM, but much more accurate.

The paper is organized as follows: The proposed direct, sparse model as well as the windowed optimization method are described in Section 8.2. Specifically, this comprises the geometric and photometric camera calibration in Section 8.2.1, the model formulation in Section 8.2.2, and the windowed optimization in Section 8.2.3. Section 8.3 describes the front-end: the part of the algorithm that performs data-selection and provides sufficiently accurate initializations for the highly non-convex optimization back-end. We provide a thorough experimental comparison to other methods in Section 8.4.1. We also evaluate the effect of important parameters and new concepts like the use of photometric calibration in Section 8.4.2. In Section 8.4.3, we analyse the effect of added photometric and geometric noise to the data. Finally, we provide a summary in Section 8.5.

8.2 Direct Sparse Model

Our direct sparse odometry is based on continuous optimization of the photometric error over a window of recent frames, taking into account a photometrically calibrated model for image formation. In contrast to existing direct methods, we jointly optimize for all involved parameters (camera intrinsics, camera extrinsics, and inverse depth values), effectively performing the photometric equivalent of windowed sparse bundle adjustment. We keep the geometry representation employed by other direct approaches, i.e., 3D points are represented as inverse depth in a reference frame (and thus have 1 degree of freedom).

Notation. Throughout the paper, bold lower-case letters ($\mathbf{x}$) represent vectors and bold upper-case letters ($\mathbf{H}$) represent matrices. Scalars will be represented by light lower-case letters (t), functions (including images) by light upper-case letters (I).

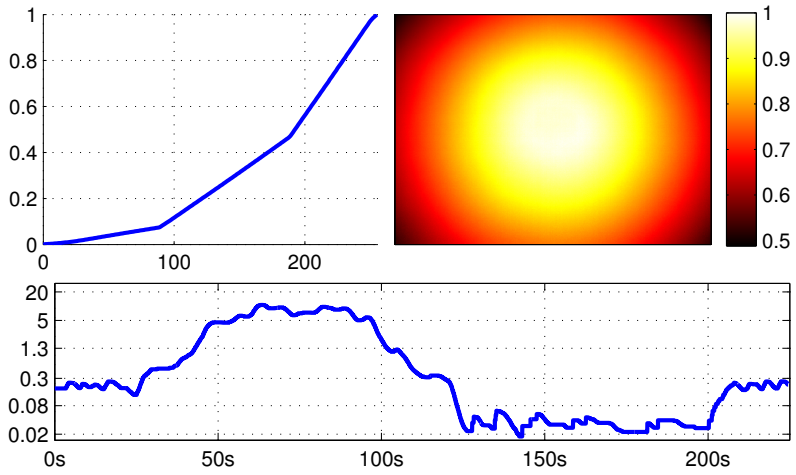


Figure 8.3: **Photometric calibration.** Top: Inverse response function G^{-1} and lens attenuation V of the camera used for Figure 8.1. Bottom: Exposure t in milliseconds for a sequence containing an indoor and an outdoor part. Note how it varies by a factor of more than 500, from 0.018 to 10.5ms. Instead of treating these quantities as unknown noise sources, we explicitly account for them in the photometric error model.

Camera poses are represented as transformation matrices $\mathbf{T}_i \in \mathrm{SE}(3)$, transforming a point from the world frame into the camera frame. Linearized pose-increments will be expressed as Lie-algebra elements $\boldsymbol{x}_i \in \mathfrak{se}(3)$, which – with a slight abuse of notation – we directly write as vectors $\boldsymbol{x}_i \in \mathbb{R}^6$. We further define the commonly used operator $\boxplus : \mathfrak{se}(3) \times \mathrm{SE}(3) \to \mathrm{SE}(3)$ using a left-multiplicative formulation, i.e.,

$$\boldsymbol{x}_i \boxplus \mathbf{T}_i := e^{\widehat{\boldsymbol{x}_i}} \cdot \mathbf{T}_i. \tag{8.1}$$

8.2.1 Calibration

The direct approach comprehensively models the image formation process. In addition to a *geometric* camera model – which comprises the function that projects a 3D point onto the 2D image – it is hence beneficial to also consider a *photometric* camera model, which comprises the function that maps real-world energy received by a pixel on the sensor (irradiance) to the respective intensity value. Note that for indirect methods this is of little benefit and hence widely ignored, as common feature extractors and descriptors are invariant (or highly robust) to photometric variations.

Geometric Camera Calibration

For simplicity, we formulate our method for the well-known pinhole camera model – radial distortion is removed in a preprocessing step. While for wide-angle cameras this does reduce the field of view, it allows comparison across methods that only

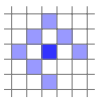


Figure 8.4: **Residual pattern.** Pattern $\mathcal{N}_\mathbf{p}$ used for energy computation. The bottom-right pixel is omitted to enable SSE-optimized processing. Note that since we have 1 unknown per point (its inverse depth), and do not use a regularizer, we require $|\mathcal{N}_\mathbf{p}| > 1$ in order for all model parameters to be well-constrained when optimizing over only two frames. Figure 8.18 shows an evaluation of how this pattern affects tracking accuracy.

implement a limited choice of camera models. Throughout this paper, we will denote projection by $\Pi_\mathbf{c} \colon \mathbb{R}^3 \to \Omega$ and back-projection with $\Pi_\mathbf{c}^{-1} \colon \Omega \times \mathbb{R} \to \mathbb{R}^3$, where $\mathbf{c}$ denotes the intrinsic camera parameters (for the pinhole model these are the focal length and the principal point). Note that analogously to [1], our approach can be extended to other (invertible) camera models, although this does increase computational demands.

Photometric Camera Calibration

We use the image formation model used in [10], which accounts for a non-linear response function $G \colon \mathbb{R} \to [0, 255]$, as well as lens attenuation (vignetting) $V \colon \Omega \to [0, 1]$. Figure 8.3 shows an example calibration from the TUM monoVO dataset. The combined model is then given by

$$I_i(\mathbf{x}) = G\Big(t_i V(\mathbf{x}) B_i(\mathbf{x})\Big), \tag{8.2}$$

where B_i and I_i are the irradiance and the observed pixel intensity in frame i, and t_i is the exposure time. The model is applied by photometrically correcting each video frame as very first step, by computing

$$I_i'(\mathbf{x}) := t_i B_i(\mathbf{x}) = \frac{G^{-1}(I_i(\mathbf{x}))}{V(\mathbf{x})}. \tag{8.3}$$

In the remainder of this paper, I_i will always refer to the photometrically corrected image I_i', except where otherwise stated.

8.2.2 Model Formulation

We define the photometric error of a point $\mathbf{p} \in \Omega_i$ in reference frame I_i, observed in a target frame I_j, as the weighted SSD over a small neighborhood of pixels. Our experiments have shown that 8 pixel, arranged in a slightly spread pattern (see Figure 8.4) give a good trade-off between computations required for evaluation, robustness to motion blur, and providing sufficient information. Note that in terms of the contained information, evaluating the SSD over such a small neighborhood

of pixels is similar to adding first- and second-order irradiance derivative constancy terms (in addition to irradiance constancy) for the central pixel. Let

$$E_{\mathbf{p}j} := \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} w_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} \big(I_i[\mathbf{p}] - b_i\big) \right\|_\gamma , \tag{8.4}$$

where $\mathcal{N}_\mathbf{p}$ is the set of pixels included in the SSD; t_i, t_j the exposure times of the images I_i, I_j; and $\|\cdot\|_\gamma$ the Huber norm. Further, $\mathbf{p}'$ stands for the projected point position of $\mathbf{p}$ with inverse depth $d_\mathbf{p}$, given by

$$\mathbf{p}' = \Pi_\mathbf{c}\Big(\mathbf{R}\ \Pi_\mathbf{c}^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}\Big), \tag{8.5}$$

with

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} := \mathbf{T}_j \mathbf{T}_i^{-1}. \tag{8.6}$$

In order to allow our method to operate on sequences without known exposure times, we include an additional affine brightness transfer function given by $e^{-a_i}(I_i - b_i)$.

In addition to using robust Huber penalties, we apply a gradient-dependent weighting $w_\mathbf{p}$ given by

$$w_\mathbf{p} := \frac{c^2}{c^2 + \|\nabla I_i(\mathbf{p})\|_2^2}, \tag{8.7}$$

which down-weights pixels with high gradient. This weighting function can be probabilistically interpreted as adding small, independent geometric noise on the projected point position $\mathbf{p}'$, and immediately marginalizing it – approximating small geometric error. To summarize, the error $E_{\mathbf{p}j}$ depends on the following variables: (1) the point's inverse depth $d_\mathbf{p}$, (2) the camera intrinsics $\mathbf{c}$, (3) the poses of the involved frames $\mathbf{T}_i, \mathbf{T}_j$, and (4) their brightness transfer function parameters a_i, b_i, a_j, b_j.

The full photometric error over all frames and points is given by

$$E_\text{photo} := \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in\text{obs}(\mathbf{p})} E_{\mathbf{p}j}. \tag{8.8}$$

where i runs over all frames $\mathcal{F}$, $\mathbf{p}$ over all points $\mathcal{P}_i$ in frame i, and j over all frames $\text{obs}(\mathbf{p})$ in which the point $\mathbf{p}$ is visible. Figure 8.5 shows the resulting factor graph: The only difference to the classical reprojection error is the additional dependency of each residual on the pose of the host frame, i.e., each term depends on *two* frames instead of only one. While this adds off-diagonal entries to the pose-pose block of the Hessian, it does not affect the sparsity pattern *after* application of the Schur complement to marginalize point parameters. The resulting system can thus be solved analogously to the indirect model. Note that the Jacobians with respect to the two frames' poses are linearly related by the adjoint of their relative pose. In

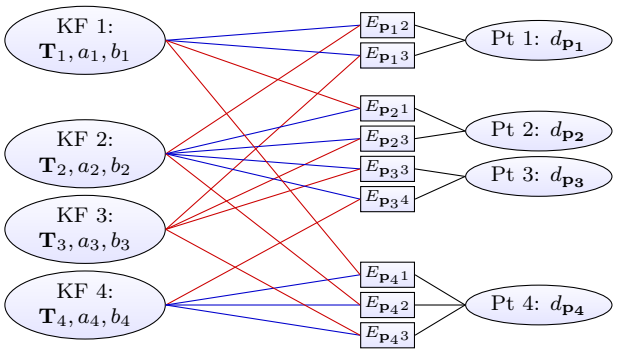


Figure 8.5: **Factor graph for the direct sparse model.** Example with four keyframes and four points; one in KF1, two in KF2, and one in KF4. Each energy term (defined in Eq. (8.4)) depends on the point's host frame (blue), the frame the point is observed in (red), and the point's inverse depth (black). Further, all terms depend on the global camera intrinsics vector $\mathbf{c}$, which is not shown.

practice, this factor can then be pulled out of the sum when computing the Hessian or its Schur complement, greatly reducing the additional computations caused by more variable dependencies.

If exposure times are known, we further add a prior pulling the affine brightness transfer function to zero:

$$E_{\text{prior}} := \sum_{i \in \mathcal{F}} \left(\lambda_a a_i^2 + \lambda_b b_i^2 \right). \tag{8.9}$$

If no photometric calibration is available, we set $t_i = 1$ and $\lambda_a = \lambda_b = 0$, as in this case they need to model the (unknown) changing exposure time of the camera. As a side-note it should be mentioned that the ML estimator for a multiplicative factor $a^* = \text{argmax}_a \sum_i (a x_i - y_i)^2$ is biased if both x_i and y_i contain noisy measurements (see [8]); causing a to drift in the unconstrained case $\lambda_a = 0$. While this generally has little effect on the estimated poses, it may introduce a bias if the scene contains only few, weak intensity variations.

Point Dimensionality. In the proposed direct model, a point is parametrized by only one parameter (the inverse depth in the reference frame), in contrast to three unknowns as in the indirect model. To understand the reason for this difference, we first note that in both cases a 3D point is in fact an arbitrarily located discrete sample on a continuous, real-world 3D surface. The difference then lies in the way this 2D location on the surface is defined. In the indirect approach, it is implicitly defined as *the point, which (projected into an image) generates a maximum in the used corner response function.* This entails that both the surface, as well as the point's location on the surface are unknowns, and need to be estimated. In our direct formulation, a point is simply defined as *the point, where the source pixel's ray hits the surface*, thus only one unknown remains. In addition to a reduced number

of parameters, this naturally enables an *inverse depth* parametrization, which – in a Gaussian frame-work – is better suited to represent uncertainty from stereo-based depth estimation, in particular for far-away points [24].

Consistency. Strictly speaking, the proposed direct sparse model does allow to use some observations (pixel values) multiple times, while others are not used at all. This is because – even though our point selection strategy attempts to avoid this by equally distributing points in space (see Section 8.3.2) – we allow point observations to overlap, and thus depend on the same pixel value(s). This particularly happens in scenes with little texture, where all points have to be chosen from a small subset of textured image regions. We however argue that this has negligible effect in practice, and – if desired – can be avoided by removing (or downweighting) observations that use the same pixel value.

8.2.3 Windowed Optimization

We follow the approach by Leutenegger et.al. [75] and optimize the total error (8.8) in a sliding window using the Gauss-Newton algorithm, which gives a good trade-off between speed and flexibility.

For ease of notation, we extend the $\boxplus$ operator as defined in (8.1) to all optimized parameters – for parameters other than SE(3) poses it denotes conventional addition. We will use $\boldsymbol{\zeta} \in \mathrm{SE}(3)^n \times \mathbb{R}^m$ to denote all optimized variables, including camera poses, affine brightness parameters, inverse depth values, and camera intrinsics. As in [75], marginalizing a residual that depends on a parameter in $\boldsymbol{\zeta}$ will fix the tangent space in which any future information (delta-updates) on that parameter is accumulated. We will denote the evaluation point for this tangent space with $\boldsymbol{\zeta}_0$, and the accumulated delta-updates by $\boldsymbol{x} \in \mathfrak{se}(3)^n \times \mathbb{R}^m$. The current state estimate is hence given by $\boldsymbol{\zeta} = \boldsymbol{x} \boxplus \boldsymbol{\zeta}_0$. Figure 8.6 visualizes the relation between the different variables.

Gauss-Newton Optimization. We compute the Gauss-Newton system as

$$\mathbf{H} = \mathbf{J}^T \mathbf{W} \mathbf{J} \qquad \text{and} \qquad \mathbf{b} = \mathbf{J}^T \mathbf{W} \mathbf{r}, \tag{8.10}$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the diagonal matrix containing the weights, $\mathbf{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathbf{J} \in \mathbb{R}^{n \times d}$ is the Jacobian of $\mathbf{r}$.

Note that each point contributes $|\mathcal{N}_p| = 8$ residuals to the energy. For notational simplicity, we will in the following consider only a single residual r_k, and the associated row of the Jacobian $\mathbf{J}_k$. During optimization – as well as when marginalizing

– residuals are always evaluated at the current state estimate, i.e.,

$$\begin{aligned} r_k &= r_k(\boldsymbol{x} \boxplus \boldsymbol{\zeta}_0) \\ &= \Big(I_j[\mathbf{p}'(\mathbf{T}_i, \mathbf{T}_j, d, \mathbf{c})] - b_j\Big) - \frac{t_j e^{a_j}}{t_i e^{a_i}}\Big(I_i[\mathbf{p}] - b_i\Big), \end{aligned} \tag{8.11}$$

where $(\mathbf{T}_i, \mathbf{T}_j, d, \mathbf{c}, a_i, a_j, b_i, b_j) := \boldsymbol{x} \boxplus \boldsymbol{\zeta}_0$ are the current state variables the residual depends on. The Jacobian $\mathbf{J}_k$ is evaluated with respect to an *additive increment* to $\boldsymbol{x}$, i.e.,

$$\mathbf{J}_k = \frac{\partial r_k((\boldsymbol{\delta} + \boldsymbol{x}) \boxplus \boldsymbol{\zeta}_0)}{\partial \boldsymbol{\delta}}. \tag{8.12}$$

It can be decomposed as

$$\mathbf{J}_k = \Bigg[\underbrace{\frac{\partial I_j}{\partial \mathbf{p}'}}_{\mathbf{J}_I} \underbrace{\frac{\partial \mathbf{p}'((\boldsymbol{\delta}+\boldsymbol{x}) \boxplus \boldsymbol{\zeta}_0)}{\partial \boldsymbol{\delta}_{\text{geo}}}}_{\mathbf{J}_{\text{geo}}}, \quad \underbrace{\frac{\partial r_k((\boldsymbol{\delta}+\boldsymbol{x}) \boxplus \boldsymbol{\zeta}_0)}{\partial \boldsymbol{\delta}_{\text{photo}}}}_{\mathbf{J}_{\text{photo}}}\Bigg], \tag{8.13}$$

where $\boldsymbol{\delta}_{\text{geo}}$ denotes the "geometric" parameters $(\mathbf{T}_i, \mathbf{T}_j, d, \mathbf{c})$, and $\boldsymbol{\delta}_{\text{photo}}$ denotes the "photometric" parameters (a_i, a_j, b_i, b_j). We employ two approximations, described below.

First, both $\mathbf{J}_{\text{photo}}$ and $\mathbf{J}_{\text{geo}}$ are evaluated at $\boldsymbol{x} = 0$. This technique is called "First Estimate Jacobians" [56, 75], and is required to maintain consistency of the system and prevent the accumulation of spurious information. In particular, in the presence of non-linear null-spaces in the energy (in our formulation absolute pose and scale), adding linearizations around different evaluation points eliminates these and thus slowly corrupts the system. In practice, this approximation is very good, since $\mathbf{J}_{\text{photo}}$, $\mathbf{J}_{\text{geo}}$ are smooth compared to the size of the increment $\boldsymbol{x}$. In contrast, $\mathbf{J}_I$ is much less smooth, but does not affect the null-spaces. Thus, it is evaluated at the current value for $\boldsymbol{x}$, i.e., at the same point as the residual r_k. We use centred differences to compute the image derivative at integer positions, which are then bilinearly interpolated.

Second, $\mathbf{J}_{\text{geo}}$ is assumed to be the same for all residuals belonging to the same point, and evaluated only for the center pixel. Again, this approximation is very good in practice – while it significantly reduces the required computations, we have not observed any notable effect on accuracy for any of the used datasets.

From the resulting linear system, an increment is computed as $\boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b}$ and added to the current state:

$$\boldsymbol{x}^{\text{new}} \leftarrow \boldsymbol{\delta} + \boldsymbol{x}. \tag{8.14}$$

Note that due to the First Estimate Jacobian approximation, a multiplicative formulation (replacing $(\boldsymbol{\delta}+\boldsymbol{x}) \boxplus \boldsymbol{\zeta}_0$ with $\boldsymbol{\delta} \boxplus (\boldsymbol{x} \boxplus \boldsymbol{\zeta}_0)$ in (8.12)) results in the exact same Jacobian, thus a multiplicative update step $\boldsymbol{x}^{\text{new}} \leftarrow \log(\boldsymbol{\delta} \boxplus e^{\boldsymbol{x}})$ is equally valid.

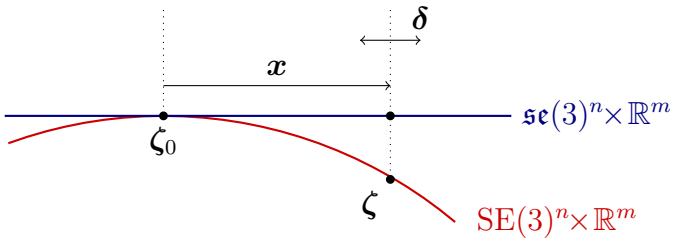


Figure 8.6: **Windowed optimization.** The red line denotes the parameter space, composed of non-Euclidean camera poses in SE(3), and the remaining euclidean parameters. The blue line corresponds to the tangent-space around $\boldsymbol{\zeta}_0$, in which we (1) accumulate the quadratic marginalization-prior on $\boldsymbol{x}$, and (2) compute Gauss-Newton steps $\boldsymbol{\delta}$. For each parameter, the tangent space is fixed as soon as that parameter becomes part of the marginalization term. Note that while we treat all parameters equally in our notation, for euclidean parameters tangent-space and parameter-space coincide.

After each update step, we update $\boldsymbol{\zeta}_0$ *for all variables that are not part of the marginalization term*, using $\boldsymbol{\zeta}_0^{\text{new}} \leftarrow \boldsymbol{x} \boxplus \boldsymbol{\zeta}_0$ and $\boldsymbol{x} \leftarrow 0$. In practice, this includes all depth values, as well as the pose of the newest keyframe. Each time a new keyframe is added, we perform up to 6 Gauss-Newton iterations, breaking early if $\boldsymbol{\delta}$ is sufficiently small. We found that – since we never start far-away from the minimum – a Levenberg-Marquad dampening (which slows down convergence) is not required.

Marginalization. When the active set of variables becomes too large, old variables are removed by marginalization using the Schur complement. Similar to [75], we drop any residual terms that would affect the sparsity pattern of $\mathbf{H}$: When marginalizing frame i, we first marginalize all points in $\mathcal{P}_i$, as well as points that have not been observed in the last two keyframes. Remaining observations of active points in frame i are dropped from the system.

Marginalization proceeds as follows: Let E' denote the part of the energy containing all residuals that depend on state variables to be marginalized. We first compute a Gauss-Newton approximation of E' around the current state estimate $\boldsymbol{\zeta} = \boldsymbol{x} \boxplus \boldsymbol{\zeta}_0$. This gives

$$\begin{aligned} & E'(\boldsymbol{x} \boxplus \boldsymbol{\zeta}_0) \\ & \approx 2(\boldsymbol{x}-\boldsymbol{x}_0)^T \mathbf{b} + (\boldsymbol{x}-\boldsymbol{x}_0)^T \mathbf{H} (\boldsymbol{x}-\boldsymbol{x}_0) + c \\ & = 2\boldsymbol{x}^T \underbrace{(\mathbf{b}-\mathbf{H}\boldsymbol{x}_0)}_{=:\mathbf{b}'} + \boldsymbol{x}^T \mathbf{H} \boldsymbol{x} + \underbrace{(c+\boldsymbol{x}_0^T \mathbf{H} \boldsymbol{x}_0 - \boldsymbol{x}_0^T \mathbf{b})}_{=:c'}, \end{aligned} \tag{8.15}$$

where $\boldsymbol{x}_0$ denotes the current value (evaluation point for $\mathbf{r}$) of $\boldsymbol{x}$. The constants c, c' can be dropped, and $\mathbf{H}, \mathbf{b}$ are defined as in (8.10-8.13). This is a quadratic function on $\boldsymbol{x}$, and we can apply the Schur complement to marginalize a subset of variables.

Written as a linear system, it becomes

$$\begin{bmatrix} \mathbf{H}_{\alpha\alpha} & \mathbf{H}_{\alpha\beta} \\ \mathbf{H}_{\beta\alpha} & \mathbf{H}_{\beta\beta} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_\alpha \\ \boldsymbol{x}_\beta \end{bmatrix} = \begin{bmatrix} \mathbf{b}'_\alpha \\ \mathbf{b}'_\beta \end{bmatrix}, \tag{8.16}$$

where β denotes the block of variables we would like to marginalize, and α the block of variables we would like to keep. Applying the Schur complement yields $\widehat{\mathbf{H}_{\alpha\alpha}}\boldsymbol{x}_\alpha = \widehat{\mathbf{b}'_\alpha}$, with

$$\widehat{\mathbf{H}_{\alpha\alpha}} = \mathbf{H}_{\alpha\alpha} - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\mathbf{H}_{\beta\alpha} \tag{8.17}$$

$$\widehat{\mathbf{b}'_\alpha} = \mathbf{b}'_\alpha - \mathbf{H}_{\alpha\beta}\mathbf{H}_{\beta\beta}^{-1}\mathbf{b}'_\beta. \tag{8.18}$$

The residual energy on $\boldsymbol{x}_\alpha$ can hence be written as

$$E'\Big(\boldsymbol{x}_\alpha \boxplus (\boldsymbol{\zeta}_0)_\alpha\Big) = 2\boldsymbol{x}_\alpha^T\widehat{\mathbf{b}'_\alpha} + \boldsymbol{x}_\alpha^T\widehat{\mathbf{H}_{\alpha\alpha}}\boldsymbol{x}_\alpha. \tag{8.19}$$

This is a quadratic function on $\boldsymbol{x}$ and can be trivially added to the full photometric error E_{photo} during all subsequent optimization and marginalization operations, replacing the corresponding non-linear terms. Note that this requires the tangent space for $\boldsymbol{\zeta}_0$ to remain the same for all variables that appear in E' during all subsequent optimization and marginalization steps.

8.3 Visual Odometry Front-End

The front end is the part of the algorithm that

- determines the sets $\mathcal{F}, \mathcal{P}_i$, and obs($\mathbf{p}$) that make up the error terms of E_{photo}. It decides which points and frames are used, and in which frames a point is visible – in particular, this includes outlier removal and occlusion detection.
- provides initializations for new parameters, required for optimizing the highly non-convex energy function E_{photo}. As a rule of thumb, a linearization of the image I is only valid in a 1-2 pixel radius; hence all parameters involved in computing $\mathbf{p}'$ should be initialized sufficiently accurately for $\mathbf{p}'$ to be off by no more than 1-2 pixels.
- decides when a point / frame should be marginalized.

As such, the front-end needs to replace many operations that in the indirect setting are accomplished by keyframe detectors (determining visibility, point selection) and initialization procedures such as RANSAC. Note that many procedures described here are specific to the monocular case. For instance, using a stereo camera makes obtaining initial depth values more straightforward, while integration of an IMU can significantly robustify – or even directly provide – a pose initialization for new frames.

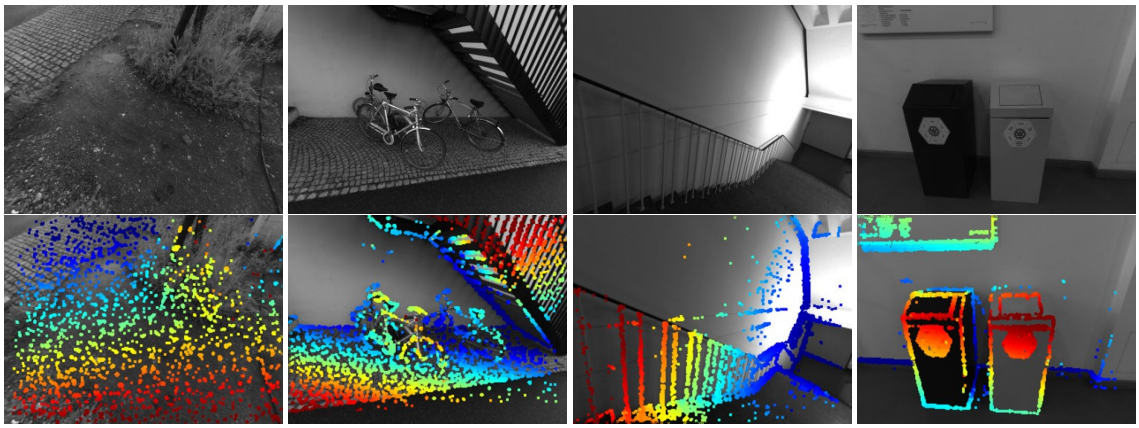


Figure 8.7: **Example depth maps used for initial frame tracking.** The top row shows the original images, the bottom row the color-coded depth maps. Since we aim at a fixed number of points in the active optimization, they become more sparse in densely textured scenes (left), while becoming similar in density to those of LSD-SLAM in scenes where only few informative image regions are available to sample from (right).

8.3.1 Frame Management

Our method always keeps a window of up to N_f active keyframes (we use $N_f = 7$). Every new frame is initially tracked with respect to these reference frames (Step 1). It is then either discarded or used to create a new keyframe (Step 2). Once a new keyframe – and respective new points – are created, the total photometric error (8.8) is optimized. Afterwards, we marginalize one or more frames (Step 3).

Step 1: Initial Frame Tracking. When a new keyframe is created, all active points are projected into it and slightly dilated, creating a semi-dense depth map. New frames are tracked with respect to only this frame using conventional two-frame direct image alignment, a multi-scale image pyramid and a constant motion model to initialize. Figure 8.7 shows some examples – we found that further increasing the density has little to no benefit in terms of accuracy or robustness, while significantly increasing runtime. Note that when down-scaling the images, a pixel is assigned a depth value if at least one of the source pixels has a depth value as in [11], significantly increasing the density on coarser resolutions.

If the final RMSE for a frame is more than twice that of the frame before, we assume that direct image alignment failed and attempt to recover by initializing with up to 27 different small rotations in different directions. This recovery-tracking is done on the coarsest pyramid level only, and takes approximately 0.5ms per try. Note that this RANSAC-like procedure is only rarely invoked, such as when the camera moves very quickly shakily. Tightly integrating an IMU would likely render this unnecessary.

Step 2: Keyframe Creation. Similar to ORB-SLAM, our strategy is to initially take many keyframes (around 5-10 keyframes per second), and sparsify them afterwards by early marginalizing redundant keyframes. We combine three criteria to determine if a new keyframe is required:

1. New keyframes need to be created as the field of view changes. We measure this by the mean square optical flow (from the last keyframe to the latest frame) $f := (\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{p} - \mathbf{p}'\|^2)^{\frac{1}{2}}$ during initial coarse tracking.

2. Camera translation causes occlusions and dis-occlusions, which requires more keyframes to be taken (even though f may be small). This is measured by the mean flow without rotation, i.e., $f_t := (\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{p} - \mathbf{p}'_t\|^2)^{\frac{1}{2}}$, where $\mathbf{p}_t$ is the warped point position with $\mathbf{R} = \mathbf{I}_{3\times 3}$.

3. If the camera exposure time changes significantly, a new keyframe should be taken. This is measured by the relative brightness factor between two frames $a := |\log(e^{a_j - a_i} t_j t_i^{-1})|$.

These three quantities can be obtained easily as a by-product of initial alignment. Finally, a new keyframe is taken if $w_f f + w_{f_t} f_t + w_a a > T_{\text{kf}}$, where w_f, w_{f_t}, w_a provide a relative weighting of these three indicators, and $T_{\text{kf}} = 1$ by default.

Step 3: Keyframe Marginalization. Our marginalization strategy is as follows (let $I_1 \dots I_n$ be the set of active keyframes, with I_1 being the newest and I_n being the oldest):

1. We always keep the latest two keyframes (I_1 and I_2).

2. Frames with less than 5% of their points visible in I_1 are marginalized.

3. If more than N_f frames are active, we marginalize the one (excluding I_1 and I_2) which maximizes a "distance score" $s(I_i)$, computed as

$$s(I_i) = \sqrt{d(i,1)} \sum_{j \in [3,n] \setminus \{i\}} (d(i,j) + \epsilon)^{-1}, \tag{8.20}$$

   where $d(i,j)$ is the Euclidean distance between keyframes I_i and I_j, and ϵ a small constant. This scoring function is heuristically designed to keep active keyframes well-distributed in 3D space, with more keyframes close to the most recent one.

A keyframe is marginalized by first marginalizing all points represented in it, and then the frame itself, using the marginalization procedure from Section 8.2.3. To preserve the sparsity structure of the Hessian, all observations of still existing points in the frame are dropped from the system. While this is clearly suboptimal (in practice about half of all residuals are dropped for this reason), it allows to efficiently optimize the energy function. Figure 8.8 shows an example of a scene, highlighting the active set of points and frames.

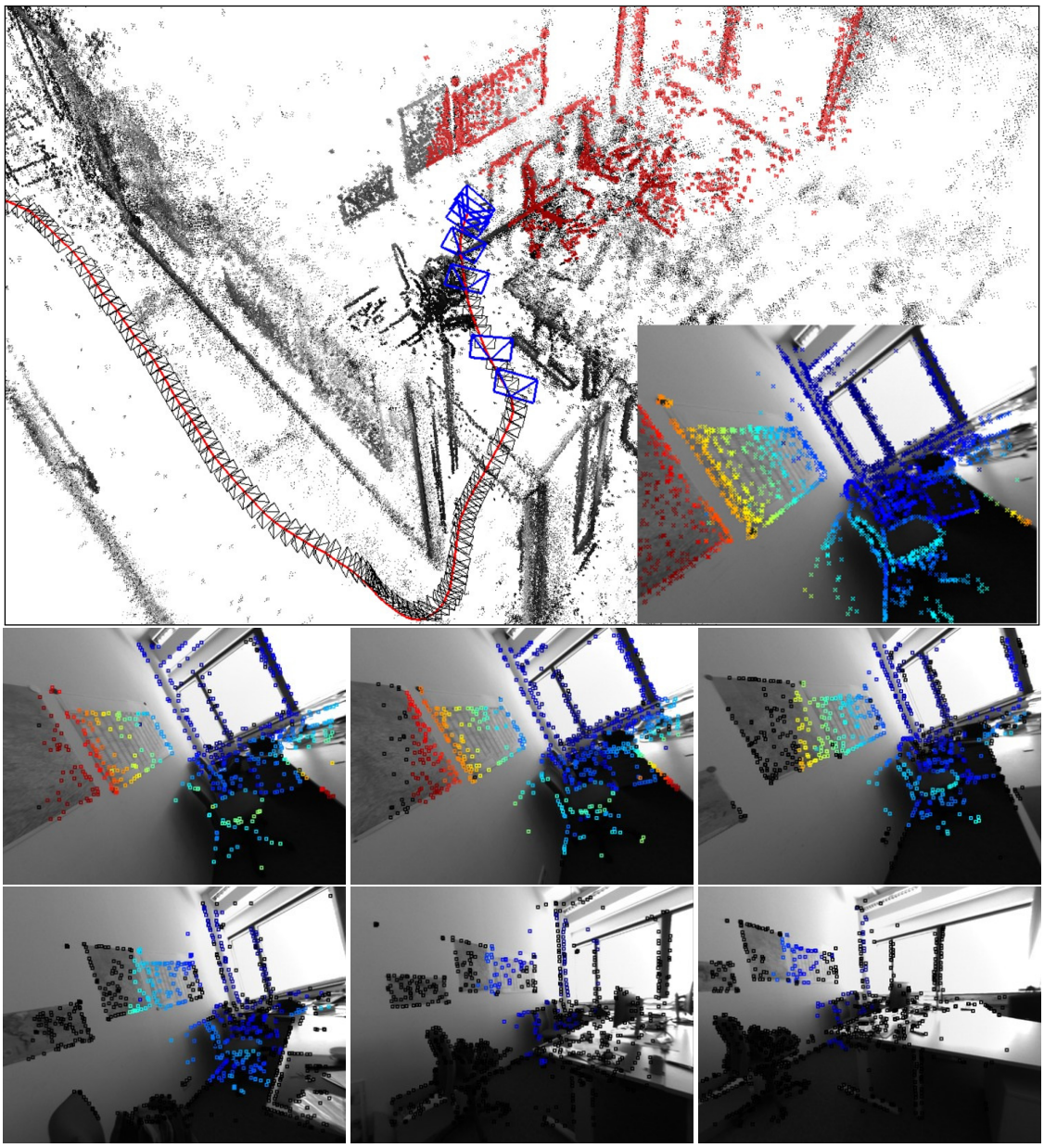


Figure 8.8: **Keyframe management.** Bottom rows: The 6 old keyframes in the optimization window, overlaid with the points hosted in them (already marginalized points are shown in black). The top image shows the full point cloud, as well as the positions of all keyframes (black camera frustums) – active points and keyframes are shown in red and blue respectively. The inlay shows the newly added keyframe, overlaid with all forward-warped active points, which will be used for initial alignment of subsequent frames.

8.3.2 Point Management

Most existing direct methods focus on utilizing as much image data as possible. To achieve this in real time, they accumulate early, sub-optimal estimates (linearizations / depth triangulations), and ignore – or approximate – correlations between different parameters. In this work, we follow a different approach, and instead heavily sub-sample data to allow processing it in real time in a joint optimization framework. In fact, our experiments show that image data is highly redundant, and the benefit of simply using *more* data points quickly flattens off. Note that in contrast to indirect methods, our direct framework still allows to *sample from across all available data*, including weakly textured or repetitive regions and edges, which does provide a real benefit (see Section 8.4).

We aim at always keeping a fixed number N_p of active points (we use $N_p = 2000$), equally distributed across space and active frames, in the optimization. In a first step, we identify N_p candidate points in each new keyframe (Step 1). Candidate points are not immediately added into the optimization, but instead are tracked individually in subsequent frames, generating a coarse depth value which will serve as initialization (Step 2). When new points need to be added to the optimization, we choose a number of candidate points (from across all frames in the optimization window) to be activated, i.e., added into the optimization (Step 3). Note that we choose N_p candidates *in each frame*, however only keep N_p active points *across all active frames combined*. This assures that we always have sufficient candidates to activate, even though some may become invalid as they leave the field of view or are identified as outliers.

Step 1: Candidate Point Selection. Our point selection strategy aims at selecting points that are (1) well-distributed in the image and (2) have sufficiently high image gradient magnitude with respect to their immediate surroundings. We obtain a region-adaptive gradient threshold by splitting the image into 32×32 blocks. For each block, we then compute the threshold as $\bar{g} + g_{\text{th}}$, where $\bar{g}$ is the median absolute gradient over all pixels in that block, and g_{th} a global constant (we use $g_{\text{th}} = 7$).

To obtain an equal distribution of points throughout the image, we split it into $d \times d$ blocks, and from each block select the pixel with largest gradient if it surpasses the region-adaptive threshold. Otherwise, we do not select a pixel from that block. We found that it is often beneficial to also include some points with weaker gradient from regions where no high-gradient points are present, capturing information from weak intensity variations originating for example from smoothly changing illumination across white walls. To achieve this, we repeat this procedure twice more, with decreased gradient threshold and block-size $2d$ and $4d$, respectively. The block-size d is continuously adapted such that this procedure generates the desired amount of points (if too many points were created it is increased for the next frame, otherwise it is decreased). Figure 8.9 shows the selected point candidates for some example

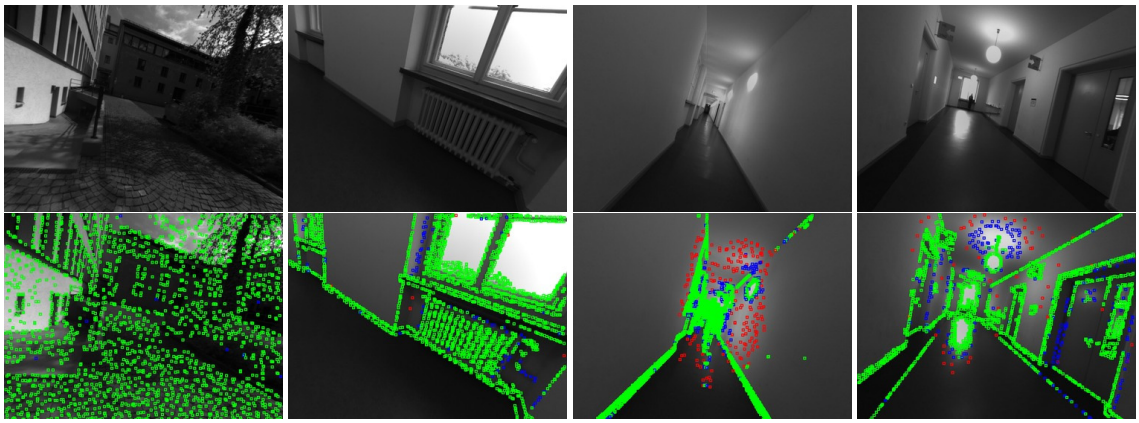


Figure 8.9: **Candidate selection.** The top row shows the original images, the bottom row shows the points chosen as candidates to be added to the map (2000 in each frame). Points selected on the first pass are shown in green, those selected on the second and third pass in blue and red respectively. Green candidates are evenly spread across gradient-rich areas, while points added on the second and third pass also cover regions with very weak intensity variations, but are much sparser.

scenes.

Step 2: Candidate Point Tracking. Point candidates are tracked in subsequent frames using a discrete search along the epipolar line, minimizing the photometric error (8.4). From the best match we compute a depth and associated variance, which is used to constrain the search interval for the subsequent frame. This tracking strategy is inspired by LSD-SLAM. Note that the computed depth only serves as *initialization* once the point is activated.

Step 3: Candidate Point Activation. After a set of old points is marginalized, new point candidates are activated to replace them. Again, we aim at maintaining a uniform spacial distribution across the image. To this end, we first project all active points onto the most recent keyframe. We then activate candidate points which – also projected into this keyframe – maximize the distance to any existing point (requiring larger distance for candidates created during the second or third block-run). Figure 8.7 shows the resulting distribution of points in a number of scenes.

Outlier and Occlusion Detection. Since the available image data generally contains much more information than can be used in real time, we attempt to identify and remove potential outliers as early as possible. First, when searching along the epipolar line during candidate tracking, points for which the minimum is not sufficiently distinct are permanently discarded, greatly reducing the number of false

matches in repetitive areas. Second, point observations for which the photometric error (8.4) surpasses a threshold are removed. The threshold is continuously adapted with respect to the median residual in the respective frame. For "bad" frames (e.g., frames that contain a lot of motion blur), the threshold will be higher, such that not all observations are removed. For good frames, in turn, the threshold will be lower, as we can afford to be more strict.

8.4 Results

In this section we will extensively evaluate our **D**irect **S**parse mono-V**O** algorithm (DSO). We both compare it to other monocular SLAM / VO methods, as well as evaluate the effect of important design and parameter choices. We use three datasets for evaluation:

(1) The **TUM monoVO dataset** [10], which provides 50 photometrically calibrated sequences, comprising 105 minutes of video recorded in dozens of different environments, indoors and outdoors. Since the dataset only provides loop-closure-ground-truth (allowing to evaluate tracking accuracy via the accumulated drift after a large loop), we evaluate using the *alignment error* (e_{align}) as defined in the respective publication.

(2) The **EuRoC MAV dataset** [22], which contains 11 stereo-inertial sequences comprising 19 minutes of video, recorded in 3 different indoor environments. For this dataset, no photometric calibration or exposure times are available, hence we omit photometric image correction and set ($\lambda_a = \lambda_b = 0$). We evaluate in terms of the absolute trajectory error (e_{ate}), which is the translational RMSE after Sim(3) alignment. For this dataset we crop the beginning of each sequence since they contain very shaky motion meant to initialize the IMU biases – we only use the parts of the sequence where the MAV is in the air.

(3) The **ICL-NUIM dataset** [52], which contains 8 ray-traced sequences comprising 4.5 minutes of video, from two indoor environments. For this dataset, photometric image correction is not required, and all exposure times can be set to $t = 1$. Again, we evaluate in terms of the absolute trajectory error (e_{ate}).

Methodology. We aim at an evaluation as comprehensive as possible given the available data, and thus run all sequences both forwards and backwards, 5 times each (to account for non-deterministic behaviour). On default settings, we run each method 10 times each. For the EuRoC MAV dataset we further run both the left and the right video separately. In total, this gives 500 runs for the TUM-monoVO dataset, 220 runs for the EuRoC MAV dataset and 80 runs for the ICL-NUIM dataset, which we run on 20 dedicated workstations. We remove the dependency on the host machine's CPU speed by *not enforcing real-time* execution, except where stated otherwise: for ORB-SLAM we play the video at 20% speed, whereas DSO

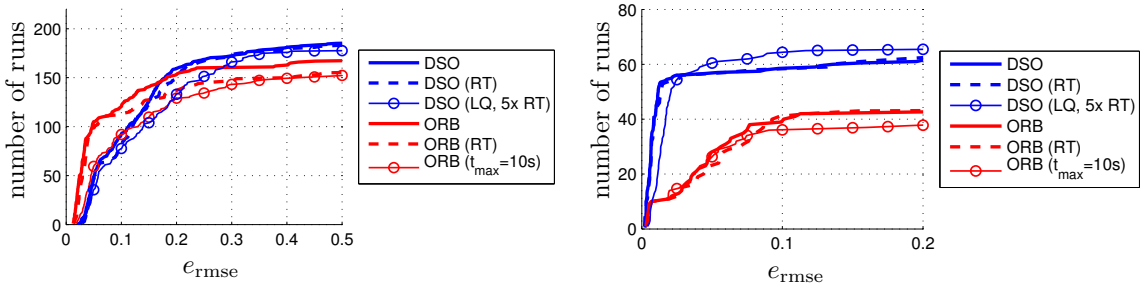


Figure 8.10: **Results on EuRoC MAV and ICL_NUIM datasets.** Translational RMSE after Sim(3) alignment. RT (dashed) denotes hard-enforced real-time execution. Further, we evaluate DSO with low settings at 5 times real-time speed, and ORB-SLAM when restricting local loop-closures to points that have been observed at least once within the last t_{max}=10s.

is run in a sequentialized, single-threaded implementation that runs approximately four times slower than real time. Note that even though we do not enforce real-time execution for most of the experiments, we use the exact same parameter settings as for the real-time comparisons.

The results are summarized in the form of *cumulative error plots* (see, e.g., Figure 8.10), which visualize for how many tracked sequences the final error was below a certain threshold; thereby showing both accuracy on sequences where a method works well, as well as robustness, i.e., on how many sequences the method does not fail. The raw tracking results for all runs – as well as scripts to compute the figures – are provided in the supplementary material[2]. Additional interesting analysis using the TUM-monoVO dataset – e.g. the influence of the camera's field of view, the image resolution or the camera's motion direction – can be found in [10].

Evaluated Methods and Parameter Settings. We compare our method to the open-source implementation of (monocular) ORB-SLAM [86]. We also attempted to evaluate against the open-source implementations of LSD-SLAM [4] and SVO [41], however both methods consistently fail on most of the sequences. A major reason for this is that they assume brightness constancy (ignoring exposure changes), while both real-world datasets used contain heavy exposure variations.

To facilitate a fair comparison and allow application of the loop-closure metric from the TUM-monoVO dataset, we disable explicit loop-closure detection and re-localization for ORB-SLAM. Note that everything else (including local and global BA) remains unchanged, still allowing ORB-SLAM to detect incremental loop-closures that can be found via the co-visibility representation alone. All parameters are set to the same value across all sequences and datasets. The only exception is

[2] `http://vision.in.tum.de/ds-vo`

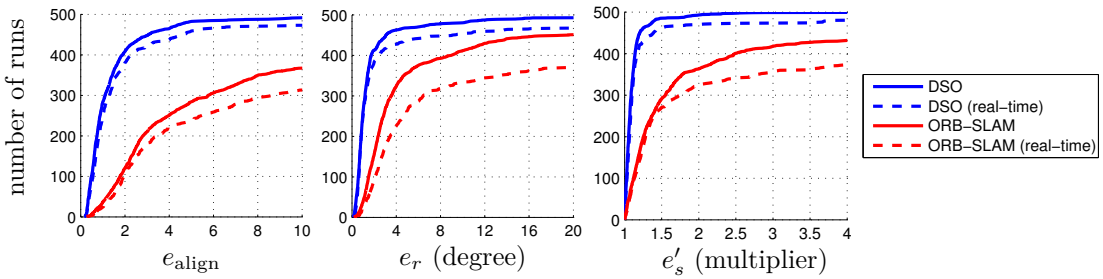


Figure 8.11: **Results on TUM-monoVO dataset.** Accumulated rotational drift e_r and scale drift e_s after a large loop, as well as the alignment error as defined in [10]. Since e_s is a multiplicative factor, we aggregate $e'_s = \max(e_s, e_s^{-1})$. The solid line corresponds to sequentialized, non-real-time execution, the dashed line to hard enforced real-time processing. For DSO, we also show results obtained at low parameter settings, running at 5 times real-time speed.

the ICL-NUIM dataset: For this dataset we set $g_{\text{th}} = 3$ for DSO, and lower the FAST threshold for ORB-SLAM to 2, which we found to give best results.

8.4.1 Quantitative Comparison

Figure 8.10 shows the absolute trajectory RMSE e_{ate} on the EuRoC MAV dataset and the ICL-NUIM dataset for both methods (if an algorithm gets lost within a sequence, we set $e_{\text{ate}} = \infty$). Figure 8.11 shows the alignment error e_{align}, as well as the rotation-drift e_r and scale-drift e_s for the TUM-monoVO dataset.

In addition to the non-real-time evaluation (bold lines), we evaluate both algorithms in a hard-enforced real-time setting on an Intel i7-4910MQ CPU (dashed lines). The direct, sparse approach clearly outperforms ORB-SLAM in accuracy and robustness both on the TUM-monoVO dataset, as well as the synthetic ICL_NUIM dataset. On the EuRoC MAV dataset, ORB-SLAM achieves a better accuracy (but lower robustness): This is due to two major reasons: (1) there is no photometric calibration available, and (2) the sequences contain many small loops or segments where the quadrocopter "back-tracks" the way it came, allowing ORB-SLAM's local mapping component to implicitly close many small and some large loops, whereas our visual odometry formulation permanently marginalizes all points and frames that leave the field of view. We can validate this by prohibiting ORB-SLAM from matching against any keypoints *that have not been observed for more than* $t_{max} = 10s$ (lines with circle markers in Figure 8.10): In this case, ORB-SLAM performs similar to DSO in terms of accuracy, but is less robust. The slight difference in robustness for DSO when running in real-time, as supposed to non-real-time, comes from the fact that for real-time execution, tracking new frames and keyframe-creation are parallelized, thus new frames are tracked on the second-latest keyframe, instead of the latest. In some rare cases – in particular during strong exposure changes – this

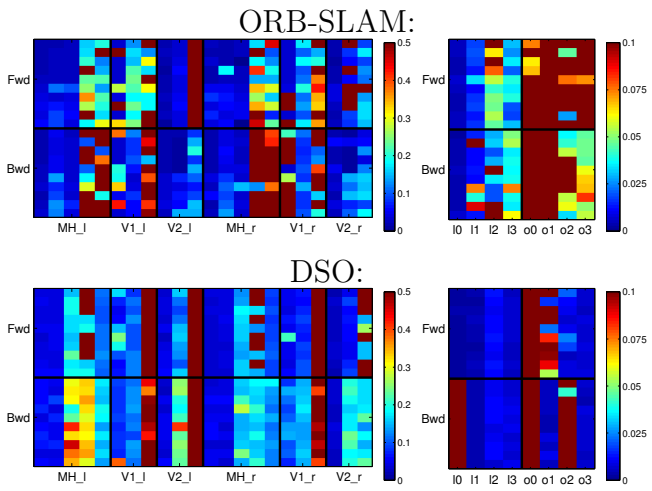


Figure 8.12: **Full evaluation result.** All error values for the EuRoC MAV dataset (left) and the ICL_NUIM dataset (right): Each square corresponds to the (color-coded) absolute trajectory error e_{ate} over the full sequence. We run each of the 11 + 8 sequences (horizontal axis) forwards ("Fwd") and backwards ("Bwd"), 10 times each (vertical axis); for the EuRoC MAV dataset we further use the left and the right image stream. Figure 8.10 shows these error values aggregated as cumulative error plot (bold, continuous lines).

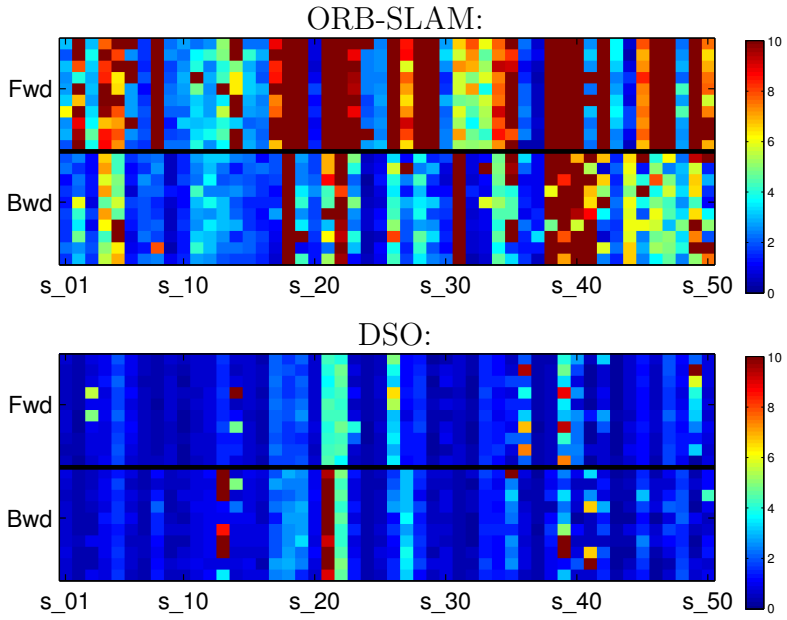


Figure 8.13: **Full evaluation result.** All error values for the TUM-monoVO dataset: Each square corresponds to the (color-coded) alignment error e_{align}, as defined in [10]. We run each of the 50 sequences (horizontal axis) forwards ("Fwd") and backwards ("Bwd"), 10 times each (vertical axis). Figure 8.11 shows all these error values aggregated as cumulative error plot (bold, continuous lines).

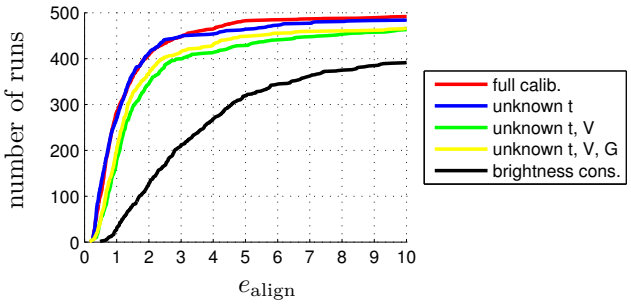


Figure 8.14: **Photometric calibration.** Errors on the TUM-monoVO dataset, when incrementally disabling photometric calibration.

causes initial image alignment to fail.

To show the flexibility of DSO, we include results when running at 5 times real-time speed[3], with reduced settings ($N_{\mathbf{p}}$=800 points, N_f=6 active frames, 424×320 image resolution, ≤ 4 Gauss-Newton iterations after a keyframe is created): Even with such extreme settings, DSO achieves very good accuracy and robustness on all three datasets.

Note that DSO is designed as a pure visual adometry while ORB-SLAM constitutes a full SLAM system, including loop-closure detection & correction and re-localization – all these additional abilities are neglected or switched off in this comparison.

8.4.2 Parameter Studies

This section aims at evaluating a number of different parameter and algorithm design choices, using the TUM-monoVO dataset.

Photometric Calibration. We analyze the influence of photometric calibration, verifying that it in fact increases accuracy and robustness: to this end, we incrementally disable the different components:

1. exposure (blue): set $t_i = 1$ and $\lambda_a = \lambda_b = 0$.
2. vignette (green): set $V(\mathbf{x}) = 1$ (and 1.).
3. response (yellow): set $G^{-1} =$ identity (and 1 – 2.).
4. brightness constancy (black): set $\lambda_a = \lambda_b = \infty$, i.e., disable affine brightness correction (and 1 – 3.).

Figure 8.14 shows the result. While known exposure times seem to have little effect on the accuracy, removing vignette and response calibration does slightly decrease the overall accuracy and robustness. Interestingly, only removing vignette calibration performs slightly worse than removing vignette and response calibration. A

[3]All images are loaded, decoded, and pinhole-rectified beforehand.

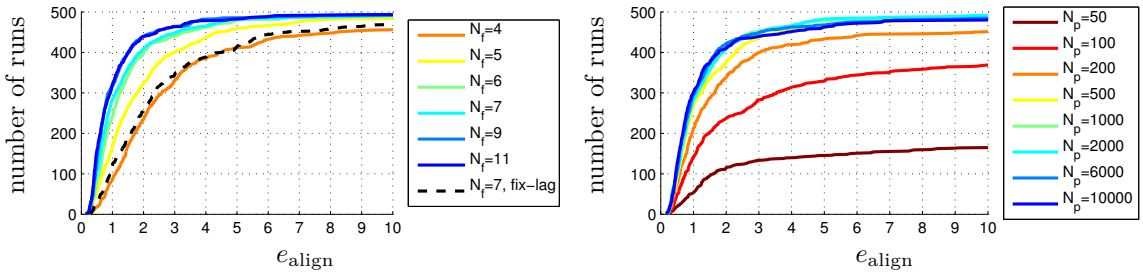


Figure 8.15: **Amount of data used.** Errors on the TUM-monoVO dataset, when changing the size of the optimization window (top) and the number of points (bottom). Using more than $N_p = 500$ points or $N_f = 7$ active frames has only marginal impact. Note that as real-time default setting, we use $N_p = 2000$ and $N_f = 7$, mainly to obtain denser reconstructions.

naïve brightness constancy assumption (as used in many other direct approaches like LSD-SLAM or SVO) clearly performs worst, since it does not account for automatic exposure changes at all.

Amount of Data. We analyze the effect of changing the amount of data used, by varying the number of active points N_p, as well as the number of frames in the active window N_f. Note that increasing N_f allows to keep more observations per point: For any point we only ever keep observations in active frames; thus the number of observations when marginalizing a point is limited to N_f (see Section 8.2.3). Figure 8.15 summarizes the result. We can observe that the benefit of simply using *more* data quickly flattens off after $N_p = 500$ points. At the same time, the number of active frames has little influence after $N_f = 7$, while increasing the runtime quadratically. We further evaluate a fixed-lag marginalization strategy (i.e., always marginalize the oldest keyframe, instead of using the proposed distancescore) as in [75]: this performs significantly worse.

Selection of Data. In addition to evaluating the effect of the number of residuals used, it is interesting to look at which data used – in particular since one of the main benefits of a direct approach is the ability to sample from all points, instead of only using corners. To this end, we vary the gradient threshold for point selection, g_{th}; the result is summarized in Figure 8.16. While there seems to be a sweet spot around $g_{\text{th}} = 7$ (if g_{th} is too large, for some scenes not enough well-distributed points are available to sample from – if it is too low, too much weight will be given to data with a low signal-to-noise ratio), the overall impact is relatively low.

More interestingly, we analyse the effect of *only using corners*, by restricting point candidates to FAST corners only. We can clearly see that only using corners significantly decreases performance. Note that for lower FAST thresholds, many false "corners" will be detected along edges, which our method can still use, in

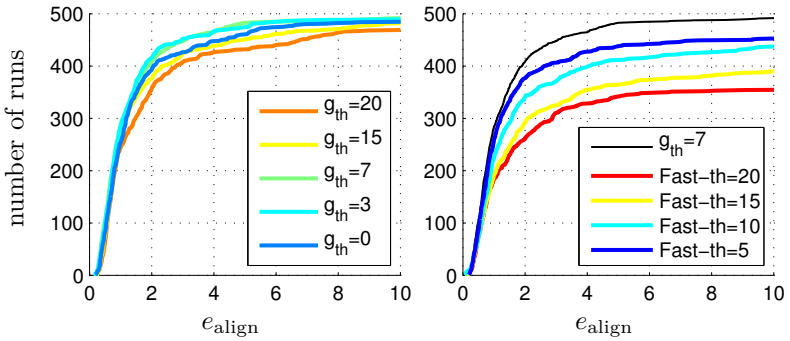


Figure 8.16: **Selection of data used.** Errors on the TUM-monoVO dataset, when changing the *type* of data used. Top: Errors for different gradient thresholds g_{th}, which seems to have a limited impact on the algorithms accuracy. Bottom: Errors when only using FAST corners, at different thresholds. Using only FAST corners significantly reduces accuracy and robustness, showing that the ability to use data from edges and weakly textured surfaces does have a real benefit.

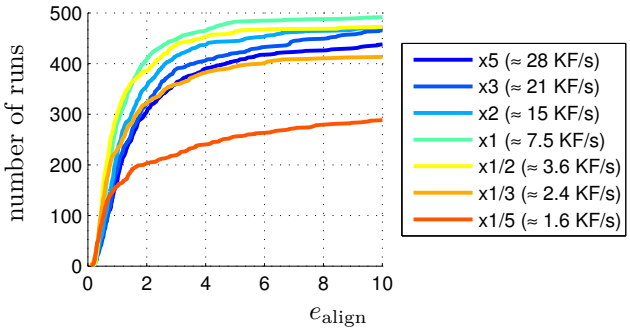


Figure 8.17: **Number of keyframes.** Errors on the TUM-monoVO dataset, when changing the number of keyframes taken via the threshold T_{kf}.

contrast to indirect methods for which such points will be outliers. In fact, ORB-SLAM achieves best performance using the default threshold of 20.

Number of Keyframes. We analyze the number of keyframes taken by varying T_{kf} (see Section 8.3.1). For each value of T_{kf} we give the resulting average number of keyframes per second; the default setting $T_{\text{kf}} = 1$ results in 8 keyframes per second, which is easily achieved in real time. The result is summarized in Figure 8.17. Taking too few keyframes (less than 4 per second) reduces the robustness, mainly in situations with strong occlusions / dis-occlusions, e.g., when walking through doors. Taking too many keyframes, on the other hand (more than 15 per second), decreases accuracy. This is because taking more keyframes causes them to be marginalized earlier (since N_f is fixed), thereby accumulating linearizations around earlier (and less accurate) linearization points.

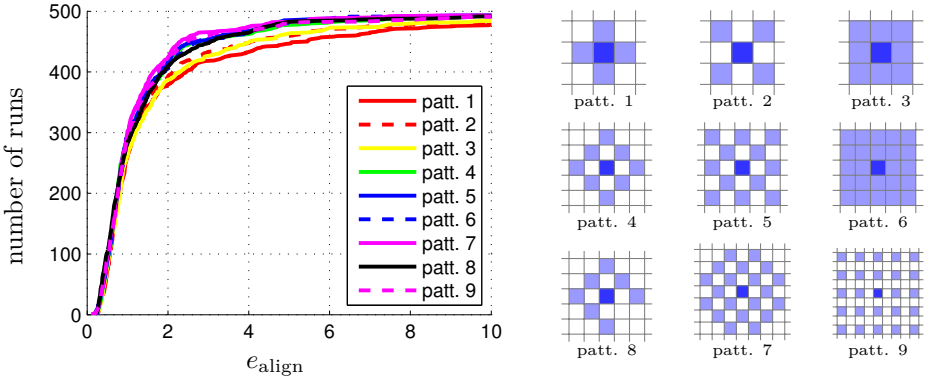


Figure 8.18: **Residual pattern.** Some of the evaluated patterns for $\mathcal{N}_{\mathbf{p}}$, and corresponding cumulative error plots. Using only a 3×3 neighborhood seems to perform slightly worse – using more than the proposed 8-pixel pattern however seems to have little benefit – at the same time, using a larger neighbourhood increases the computational demands.

Residual Pattern. We test different residual patterns for $\mathcal{N}_{\mathbf{p}}$, covering smaller or larger areas. The result is shown in Figure 8.18.

8.4.3 Geometric vs. Photometric Noise Study

The fundamental difference between the proposed direct model and the indirect model is the noise assumption. *The direct approach models photometric noise*, i.e., additive noise on pixel intensities. In contrast, *the indirect approaches models geometric noise*, i.e., additive noise on the (u, v)-position of a point in the image plane, assuming that keypoint descriptors are robust to photometric noise. It therefore comes at no surprise that the indirect approach is significantly more robust to geometric noise in the data. In turn, the direct approach performs better in the presence of strong photometric noise – which keypoint-descriptors (operating on a purely local level) fail to filter out. We verify this by analyzing tracking accuracy on the TUM-monoVO dataset, when artificially adding (a) geometric noise, and (b) photometric noise to the images.

Geometric Noise. For each frame, we separately generate a low-frequency random flow-map $N_g \colon \Omega \to \mathbb{R}^2$ by up-sampling a 3×3 grid filled with uniformly distributed random values from $[-\delta_g, \delta_g]^2$ (using bicubic interpolation). We then perturb the original image by shifting each pixel $\mathbf{x}$ by $N_g(\mathbf{x})$:

$$I'_g(\mathbf{x}) := I(\mathbf{x} + N_g(\mathbf{x})). \tag{8.21}$$

This procedure simulates noise originating from (unmodeled) rolling shutter or inaccurate geometric camera calibration. Figure 8.19 visualizes an example of the

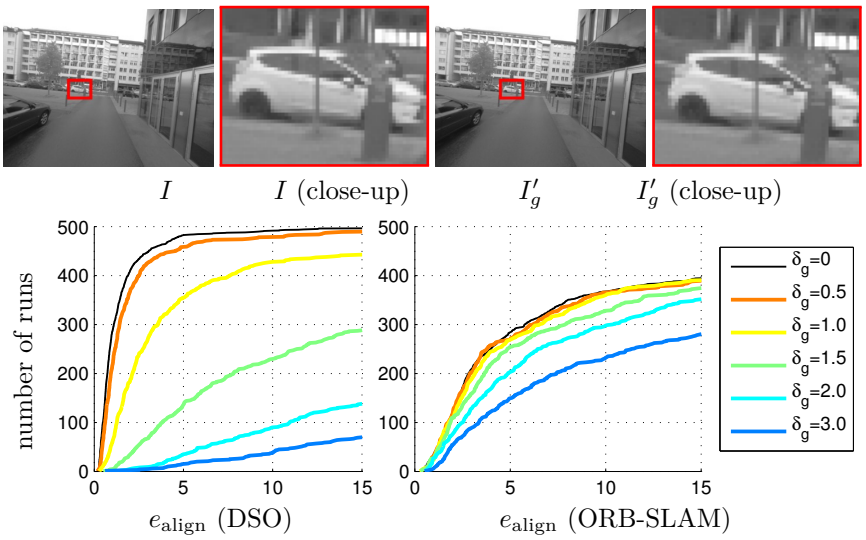


Figure 8.19: **Geometric noise.** Effect of applying low-frequency geometric noise to the image, simulating geometric distortions such as a rolling shutter. The top row shows an example image with $\delta_g = 2$. While the effect is hardly visible to the human eye (observe that the close-up is slightly shifted), it has a severe impact on SLAM accuracy, in particular when using a direct model. Note that the distortion caused by a standard rolling shutter camera easily surpasses $\delta_g = 3$.

resulting noise pattern, as well as the accuracy of ORB-SLAM and DSO for different values of δ_g. As expected, we can clearly observe how DSO's performance quickly deteriorates with added geometric noise, whereas ORB-SLAM is much less affected. This is because the first step in the indirect pipeline – keypoint detection and extraction – is not affected by low-frequent geometric noise, as it operates on a purely local level. The second step then optimizes a geometric noise model – which not surprisingly deals well with geometric noise. In the direct approach, in turn, geometric noise is not modeled, and thus has a much more severe effect – in fact, for $\delta_g > 1.5$ there likely exists no state for which all residuals are within the validity radius of the linearization of I; thus optimization fails entirely (which can be alleviated by using a coarser pyramid level). Note that this result also suggests that the proposed direct model is more susceptible to inaccurate intrinsic camera calibration than the indirect approach – in turn, it may benefit more from accurate, non-parametric intrinsic calibration.

Photometric Noise. For each frame, we separately generate a high-frequency random blur-map $N_p \colon \Omega \to \mathbb{R}^2$ by up-sampling a 300×300 grid filled with uniformly distributed random values in $[-\delta_p, \delta_p]^2$. We then perturb the original image by

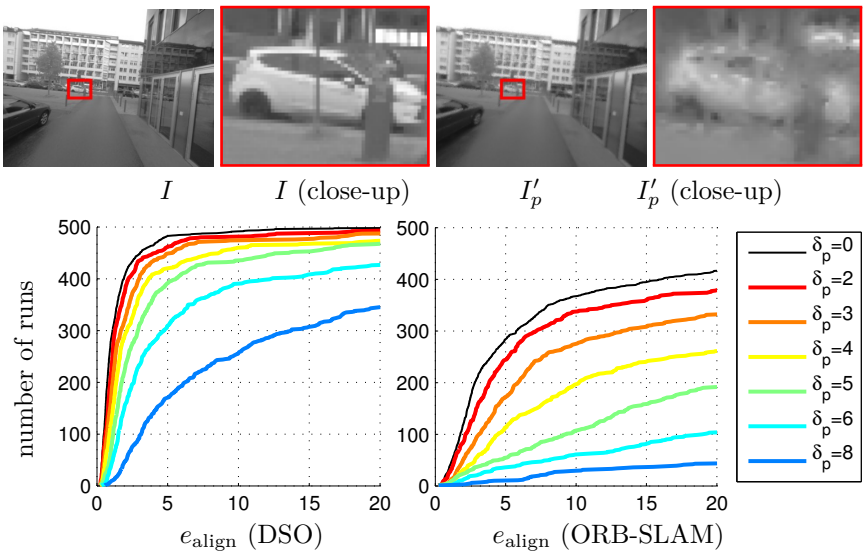


Figure 8.20: **Photometric noise.** Effect of applying high-frequent, non-isotropic blur to the image, simulating photometric noise. The top row shows an example image with $\delta_p = 6$, the effect is clearly visible. Since the direct approach models a photometric error, it is more robust to this type of noise than indirect methods.

adding anisotropic blur with standard deviation $N_p(\mathbf{x})$ to pixel $\mathbf{x}$:

$$I'_p(\mathbf{x}) := \int_{\mathbb{R}^2} \phi(\boldsymbol{\delta}; N_p(\mathbf{x})^2) I(\mathbf{x} + \boldsymbol{\delta}) \quad d\boldsymbol{\delta}, \tag{8.22}$$

where $\phi(\cdot; N_p(\mathbf{x})^2)$ denotes a 2D Gaussian kernel with standard deviation $N_p(\mathbf{x})$. Figure 8.20 shows the result. We can observe that DSO is slightly more robust to photometric noise than ORB-SLAM – this is because (purely local) keypoint matching fails for high photometric noise, whereas a joint optimization of the photometric error better overcomes the introduced distortions.

To summarize: While the direct approach outperforms the indirect approach on well-calibrated data, it is ill-suited in the presence of strong geometric noise, e.g., originating from a rolling shutter or inaccurate intrinsic calibration. In practice, this makes the indirect model superior for smartphones or off-the-shelf webcams, since these were designed to capture videos for human consumption – prioritizing resolution and light-sensitivity over geometric precision. In turn, the direct approach offers superior performance on data captured with dedicated cameras for machine-vision, since these put more importance on geometric precision, rather than capturing appealing images for human consumption. Note that this can be resolved by tightly integrating the rolling shutter into the model, as done, e.g., in [64, 76].

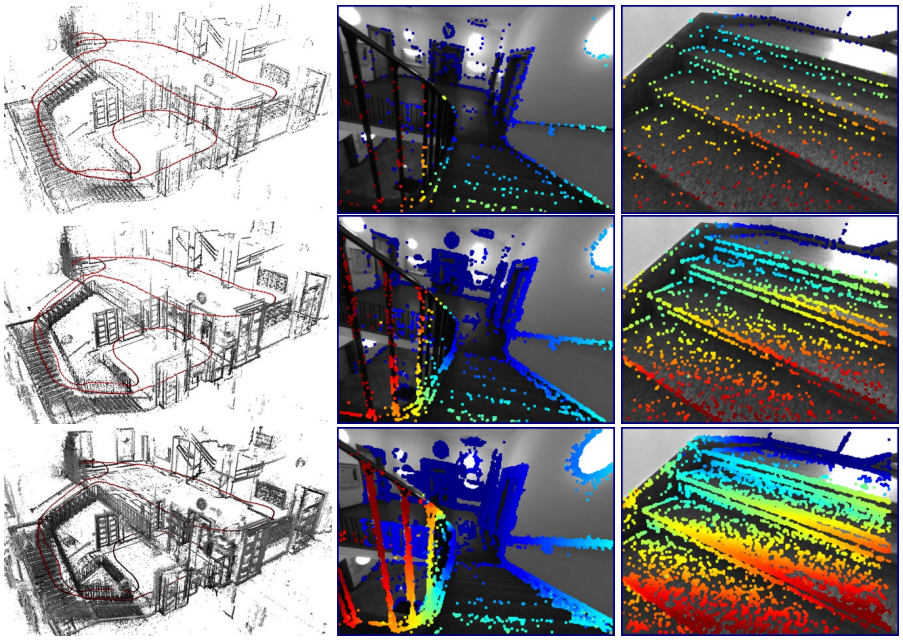


Figure 8.21: **Point density.** 3D point cloud and some coarse depth maps, i.e., the most recent keyframe with all N_p active points projected into it) for N_p=500 (top), N_p=2000 (middle), and N_p=10000 (bottom).

8.4.4 Qualitative Results

In addition to accurate camera tracking, DSO computes 3D points on all gradient-rich areas, including edges – resulting in point-cloud reconstructions similar to the semi-dense reconstructions of LSD-SLAM. The density then directly corresponds to how many points we keep in the active window N_p. Figure 8.21 shows some examples.

Figure 8.22 shows three more scenes (one from each dataset), together with some corresponding depth maps. Note that our approach is able to track through scenes with very little texture, whereas indirect approaches fail. All reconstructions shown are simply accumulated from the odometry, without integrating loop-closures. See the supplementary video for more qualitative results.

8.5 Conclusion

We have presented a novel *direct* and *sparse* formulation for Structure from Motion. It combines the benefits of direct methods (seamless ability to use & reconstruct all points instead of only corners) with the flexibility of sparse approaches (efficient, joint optimization of all model parameters). This is possible in real time by omitting the geometric prior used by other direct methods, and instead evaluat-

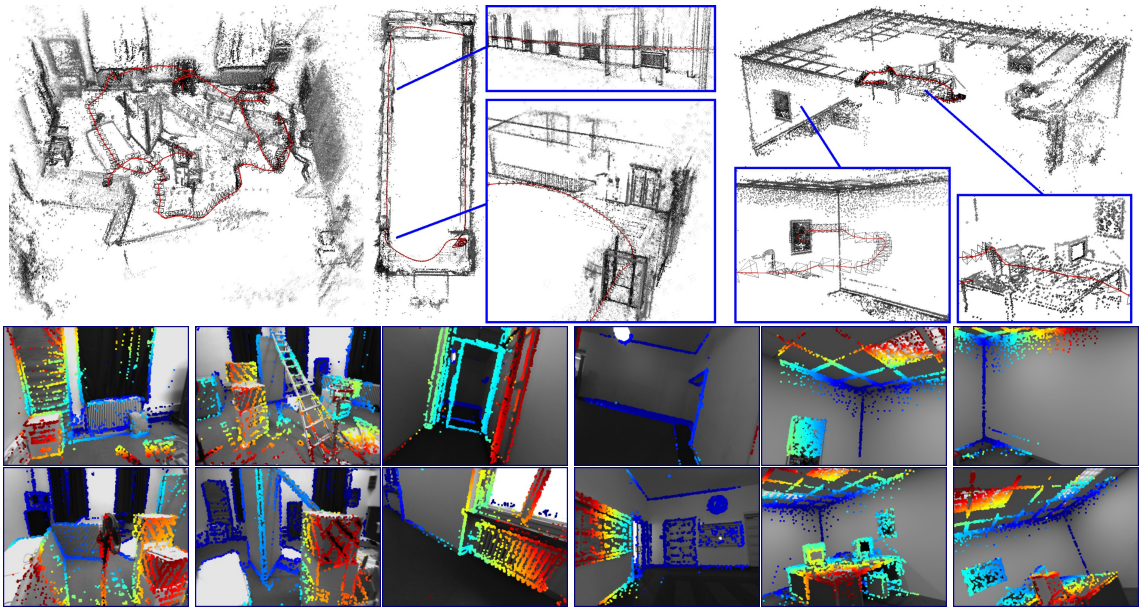


Figure 8.22: **Qualitative examples.** One scene from each dataset (left to right: *V2_01_easy* [22], *seq_38* [10] and *office_1* [52]), computed in real time with default settings. The bottom shows some corresponding (sparse) depth maps – some scenes contain very little texture, making them very challenging for indirect approaches.

ing the photometric error for each point over a small neighborhood of pixels, to well-constrain the overall problem. Furthermore, we incorporate full photometric calibration, completing the intrinsic camera model that traditionally only reflects the geometric component of the image formation process.

We have implemented our direct & sparse model in the form of a monocular visual odometry algorithm (DSO), incrementally marginalizing / eliminating old states to maintain real-time performance. To this end we have developed a front-end that performs data-selection and provides accurate initialization for optimizing the highly non-convex energy function. Our comprehensive evaluation on several hours of video shows the superiority of the presented formulation relative to state-of-the-art indirect methods. We furthermore present an exhaustive parameter study, indicating that (1) simply using more data does not increase tracking accuracy (although it makes the 3D models denser), (2) using all points instead of only corners does provide a real gain in accuracy and robustness, and (3) incorporating photometric calibration does increase performance, in particular compared to the basic "brightness constancy" assumption. We have also shown experimentally that the indirect approach – modelling a geometric error – is much more robust to geometric noise, e.g., originating from poor intrinsic camera calibration or rolling shutter. The performance of the direct model, in turn, quickly deteriorates in the presence of such noise. In practice, this means that the indirect approach will likely perform better on data captured by off-the-shelf cameras (unless the rolling shutter is mod-

eled explicitly), while the potential of direct methods only becomes apparent when also considering the used sensor.

Since the structure of the proposed direct sparse energy formulation is the same as that of indirect methods, it can be integrated with other optimization frameworks like (double-windowed) bundle adjustment [108] or incremental smoothing and mapping [60]. The main challenge here is the greatly increased degree of non-convexity compared to the indirect model, which originates from the inclusion of the image in the error function – this is likely to restrict the use of our model to video processing.

Chapter 9

A Photometrically Calibrated Benchmark for Monocular Visual Odometry

Authors	Jakob Engel[1]	engelj@in.tum.de
	Vladyslav Usenko[1]	usenko@in.tum.de
	Daniel Cremers[1]	cremers@tum.de
	[1] Technical University Munich	

Publication	**A Photometrically Calibrated Benchmark For Monocular Visual Odometry**. J. Engel, V. Usenko, D. Cremers Published as *arXiv:1607.02565 preprint*, July 2016.	

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

Abstract. We present a dataset for evaluating the tracking accuracy of monocular visual odometry and SLAM methods. It contains 50 real-world sequences comprising more than 100 minutes of video, recorded across dozens of different environments – ranging from narrow indoor corridors to wide outdoor scenes. All sequences contain mostly exploring camera motion, starting and ending at the same position. This allows to evaluate tracking accuracy via the accumulated drift from start to end, without requiring ground truth for the full sequence. In contrast to existing datasets, all sequences are photometrically calibrated. We provide exposure times for each frame as reported by the sensor, the camera response function, and dense lens attenuation factors. We also propose a novel, simple approach to non-parametric vignette calibration, which requires minimal set-up and is easy to reproduce. Finally, we thoroughly evaluate two existing methods (ORB-SLAM [86] and DSO [3]) on the dataset, including an analysis of the effect of image resolution, camera field of view, and the camera motion direction.

9.1 Introduction

Structure from Motion or Simultaneous Localization and Mapping (SLAM) has become an increasingly important topic, since it is a fundamental building block for many emerging technologies – from autonomous cars and quadrocopters to virtual and augmented reality. In all these cases, sensors and cameras built into the hardware are designed to produce data well-suited for computer vision algorithms, instead of capturing images optimized for human viewing. In this paper we present a new monocular visual odometry (VO) / SLAM evaluation benchmark, that attempts to resolve two current issues:

Sensors Intrinsics. Many existing methods are designed to operate on, and are evaluated with, data captured by commodity cameras without taking advantage of knowing – or even being able to influence – the full image formation pipeline. Specifically, methods are designed to be robust to (assumed unknown) automatic exposure changes, non-linear response functions (gamma correction), lens attenuation (vignetting), de-bayering artifacts, or even strong geometric distortions caused by a rolling shutter. This is particularly true for modern keypoint detectors and descriptors, which are robust or invariant to arbitrary monotonic brightness changes. However, recent direct methods as well attempt to compensate for automatic exposure changes, e.g., by optimizing an affine mapping between brightness values in different images [5]. Most direct methods however simply assume constant exposure time [4, 65, 88, 93].

While this is the only way to evaluate on existing datasets and with off-the-shelf

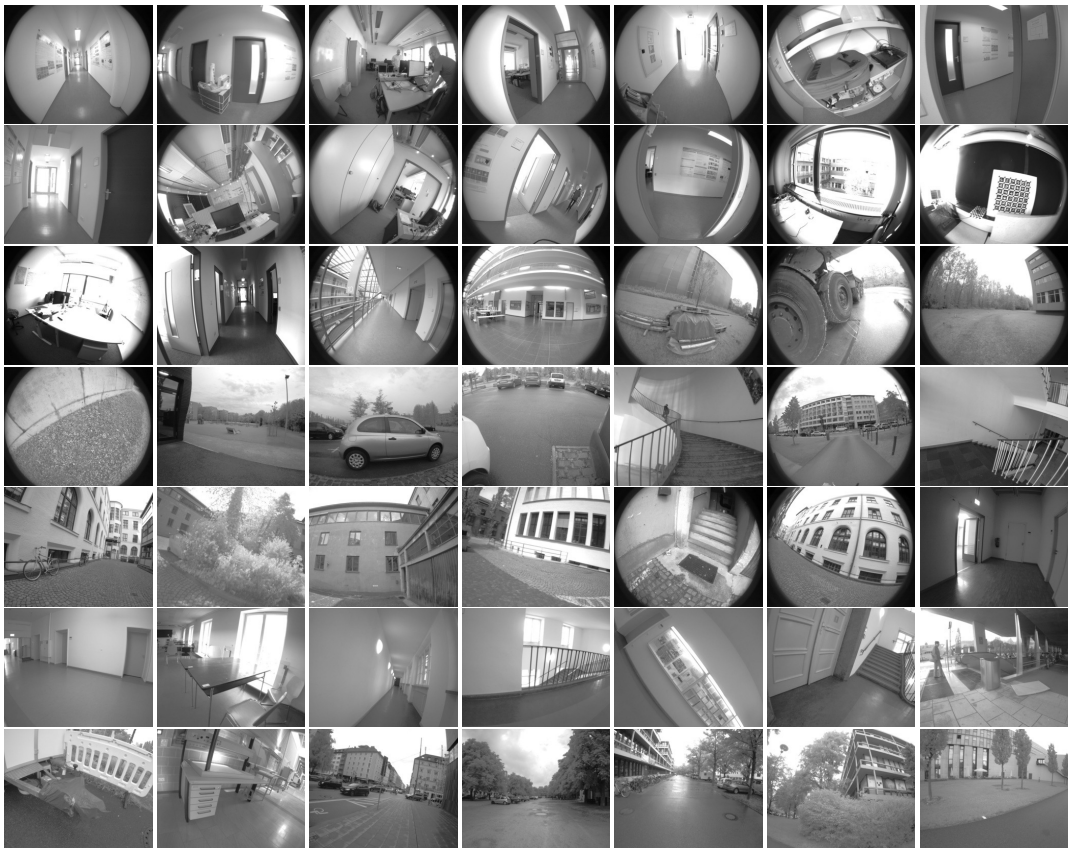


Figure 9.1: **The TUM monoVO dataset.** A single frame from each of the 50 sequences. Note the wide variety of covered environments. The full dataset contains over 190'000 frames (105 minutes) of video taken with two different lenses, exposure times for each frame, and a photometric calibration including camera response and vignetting.

commodity cameras (which often do not allow to either read or set parameters like the exposure time), we argue that for the above-mentioned use cases, this ultimately is the wrong approach: sensors – including cameras – can, and will be designed to fit the needs of the algorithms processing their data. In turn, algorithms should take full advantage of the sensor's capabilities and incorporate knowledge about the sensor design. Simple examples are image exposure time and hardware gamma correction, which are intentionally built into the camera to produce better images. Instead of treating them as unknown noise factors and attempt to correct for them afterwards, they can be treated as feature that can be modelled by, and incorporated into the algorithm – rendering the obtained data more meaningful.

Benchmark Size. SLAM and VO are complex, very non-linear estimation problems and often minuscule changes can greatly affect the outcome. To obtain a meaningful comparison between different methods and to avoid manual overfitting to specific environments or motion patterns (except for cases where this is specifically desired), algorithms should be evaluated on large datasets in a wide variety of scenes. However, existing datasets often contain only a limited number of environments. The major reason for this is that accurate ground truth acquisition is challenging, in particular if a wide range of different environments is to be covered: GPS/INS is limited in accuracy and only possible in outdoor environments with adequate GPS reception. External motion capture systems on the other hand are costly and time-consuming to set up, and can only cover small (indoor) environments.

The dataset published in this paper attempts to tackle these two issues. First, it contains frame-wise exposure times as reported by the sensor, as well as accurate calibrations for the sensors response function and lens vignetting, which enhances the performance particularly of direct approaches. Second, it contains 50 sequences with a total duration of 105 minutes (see Figure 9.1), captured in dozens of different environments. To make this possible, we propose a new evaluation methodology which does not require ground truth from external sensors – instead, tracking accuracy is evaluated by measuring the accumulated drift that occurs after a large loop. We further propose a novel, straight-forward approach to calibrate a non-parametric response function and vignetting map with minimal set up required, and without imposing a parametric model which may not suit all lenses / sensors.

9.1.1 Related Work: Datasets

There exists a number of datasets that can be used for evaluating monocular SLAM or VO methods. We will here list the most commonly used ones.

KITTI [45]: 21 stereo sequences recorded from a driving car, motion patterns and environments are limited to forward-motion and street-scenes. Images are pre-rectified, raw sensor measurements or calibration datasets are not available. The benchmark contains GPS-INS ground truth poses for all frames.

EUROC MAV [22]: 11 stereo-inertial sequences from a flying quadrocopter in three different indoor environments. The benchmark contains ground truth poses for all frames, as well as the raw sensor data and respective calibration datasets.

TUM RGB-D [114]: 89 sequences in different categories (not all meant for SLAM) in various environments, recorded with a commodity RGB-D sensor. They contain strong motion blur and rolling-shutter artifacts, as well as degenerate (rotation-only) motion patterns that cannot be tracked well from monocular odometry alone. Sequences are pre-rectified, the raw sensor data is not available. The benchmark contains ground truth poses for all sequences.

ICL-NUIM [52]: 8 ray-traced RGB-D sequences from 2 different environments. It provides a ground truth intrinsic calibration; a photometric calibration is not required, as the virtual exposure time is constant. Some of the sequences contain degenerate (rotation-only) motion patterns that cannot be tracked well from a monocular camera alone.

9.1.2 Related Work: Photometric Calibration

Many approaches exist to calibrate and remove vignetting artefacts and account for non-linear response functions. Early work focuses on image stitching and mosaicking, where the required calibration parameters need to be estimated from a small set of overlapping images [48] [67] [21]. Since the available data is limited, such methods attempt to find low-dimensional (parametric) function representations, like radially symmetric polynomial representations for vignetting. More recent work [63, 99] has shown that such representations may not be sufficiently expressive to capture the complex nature of real-world lenses and hence advocate non-parametric – dense – vignetting calibration. In contrast to [63, 99], our formulation however does not require a "uniformly lit white paper", simplifying the required calibration set-up.

For response function estimation, a well-known and straight-forward method is that of Debevec and Malik [32], which – like our approach – recovers a 2^8-valued lookup table for the inverse response from two or more images of a static scene at different exposures.

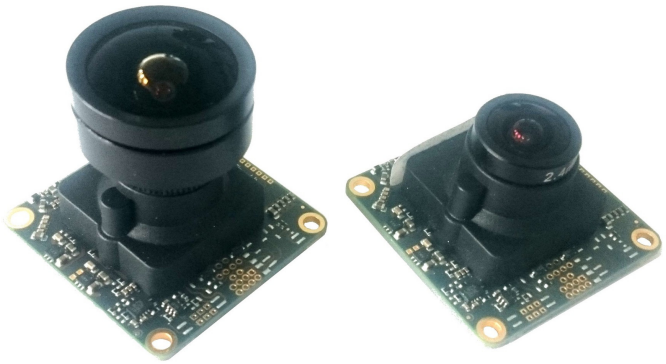


Figure 9.2: **Cameras used to capture the dataset.** Left: narrow lens (98° × 79° non-rectified field of view), right: wide lens (148° × 122° non-rectified field of view).

9.1.3 Paper Outline

The paper is organized as follows: In Section 9.2, we first describe the hardware set-up, followed by both the used distortion model (geometric calibration) in 9.2.2, as well as photometric calibration (vignetting and response function) and the proposed calibration procedure in 9.2.3. Section 9.3 describes the proposed loop-closure evaluation methodology and respective error measures. Finally, in Section 9.4, we give extensive evaluation results of two state-of-the-art monocular SLAM / VO systems ORB-SLAM [86] and Direct Sparse odometry (DSO) [3]. We further show some exemplary image data and describe the dataset contents. In addition to the dataset, we publish all code and raw evaluation data as open-source.

9.2 Calibration

We provide both a standard camera intrinsic calibration using the FOV camera model, as well as a photometric calibration, including vignetting and camera response function.

9.2.1 Hardware

The cameras used for recording the sequences are uEye UI-3241LE-M-GL monochrome, global shutter CMOS cameras from IDS. They are capable of recording 1280×1024 videos at up to 60fps. Sequences are recorded at different framerates ranging from 20fps to 50fps with jpeg-compression. For some sequences hardware gamma correction is enabled and for some it is disabled. We use two different lenses (Lensagon BM2420 with a field of view of 148° × 122°, as well as a Lensagon BM4018S118 with a field of view of 98° × 79°), as shown in Figure 9.2. Figure 9.1 shows a number of example images from the dataset.

9.2.2 Geometric Intrinsic Calibration

We use the pinhole camera model in combination with a FOV distortion model, since it is well-suited for the used fisheye lenses. For a given 3D point $(x, y, z) \in \mathbb{R}^3$ in the camera coordinate system, the corresponding point in the image $(u_d, v_d) \in \Omega$ is computed by first applying a pinhole projection followed by radial distortion and conversion to pixel coordinates

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \frac{1}{r_u \omega} \arctan\left(2r_u \tan\left(\frac{\omega}{2}\right)\right) \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{9.1}$$

where $r_u := \sqrt{(\frac{x}{z})^2 + (\frac{y}{z})^2}$ is the radius of the point in normalized image coordinates.

A useful property of this model is the existence of a closed-form inverse: for a given point the image (u_d, v_d) and depth d, the corresponding 3D point can be computed by first converting it back to normalized image coordinates

$$\begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix} = \begin{bmatrix} (u_d - c_x) f_x^{-1} \\ (v_d - c_y) f_y^{-1} \end{bmatrix}, \tag{9.2}$$

then removing radial distortion

$$\begin{bmatrix} \tilde{u}_u \\ \tilde{v}_u \end{bmatrix} = \frac{\tan(r_d \omega)}{2r_d \tan\frac{\omega}{2}} \begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix}, \tag{9.3}$$

where $r_d := \sqrt{\tilde{u}_d^2 + \tilde{v}_d^2}$. Afterwards the point is back-projected using $(x, y, z) = d(\tilde{u}_u, \tilde{v}_u, 1)$. We use the camera calibrator provided with the open-source implementation of PTAM [69] to calibrate the parameters $[f_x, f_y, c_x, c_y, \omega]$ from a number of checkerboard images.

9.2.3 Photometric Calibration

We provide photometric calibrations for all sequences. We calibrate the camera response function G, as well as pixel-wise attenuation factors $V \colon \Omega \to [0, 1]$ (vignetting). Without known irradiance, both G and V are only observable up to a scalar factor. The combined image formation model is then given by

$$I(\mathbf{x}) = G\Big(tV(\mathbf{x})B(\mathbf{x})\Big), \tag{9.4}$$

where t is the exposure time, B the irradiance image (up to a scalar factor), and I the observed pixel value. As a shorthand, we will use $U := G^{-1}$ for the inverse response function.

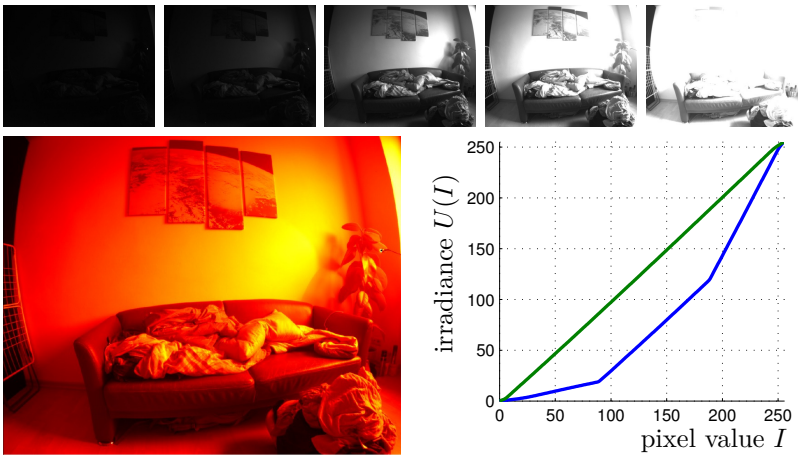


Figure 9.3: **Response calibration.** The top row shows five out of over 1000 images of the same scene at different exposures, used for calibration. The bottom-left image shows the estimated log-irradiance $\log(B')$; the bottom-right image shows the estimated inverse response U, with enabled hardware gamma correction (blue) and without (green).

Response Calibration

We first calibrate the camera response function from a sequence of images taken of a static scene with different (known) exposure time. The content of the scene is arbitrary – however to well-constrain the problem, it should contain a wide range of gray values. We first observe that in a static scene, the attenuation factors can be absorbed in the irradiance image, i.e.,

$$I(\mathbf{x}) = G\Big(tB'(\mathbf{x})\Big), \tag{9.5}$$

with $B'(\mathbf{x}) := V(\mathbf{x})B(\mathbf{x})$. Given a number of images I_i, corresponding exposure times t_i and a Gaussian white noise assumption on $U(I_i(\mathbf{x}))$, this leads to the following Maximum-Likelihood energy formulation

$$E(U, B') = \sum_i \sum_{\mathbf{x}\in\Omega} \Big(U\Big(I_i(\mathbf{x})\Big) - t_i B'(\mathbf{x})\Big)^2. \tag{9.6}$$

For overexposed pixels, U is not well defined, hence they are removed from the estimation. We now minimize (9.6) alternatingly for U and B'. Note that fixing either U or B' de-couples all remaining values, such that minimization becomes trivial:

$$U(k)^* = \underset{U(k)}{\operatorname{argmin}}\, E(U, B') = \frac{\sum_{\Omega_k} t_i B'(\mathbf{x})}{|\Omega_k|} \tag{9.7}$$

$$B'(\mathbf{x})^* = \underset{B'(\mathbf{x})}{\operatorname{argmin}}\, E(U, B') = \frac{\sum_i t_i U\Big(I_i(\mathbf{x})\Big)}{\sum_i t_i^2}, \tag{9.8}$$

where $\Omega_k := \{i, \mathbf{x} | I_i(\mathbf{x}) = k\}$ is the set of all pixels in all images that have intensity k. Note that the resulting U may not be monotonic – which is a pre-requisite for invertibility. In this case it needs to be smoothed or perturbed; however for all our calibration datasets this does not happen. The value for $U(255)$ is never observed since overexposed pixels are removed, and needs to be extrapolated from the adjacent values. After minimization, U is scaled such that $U(255) = 255$ to disambiguate the unknown scalar factor. Figure 9.3 shows the estimated values for one of the calibration sequences.

Note on Observability. In contrast to [32], we do not employ a smoothness prior on U – instead, we use large amounts of data (1000 images covering 120 different exposure times, ranging from 0.05ms to 20ms in multiplicative increments of 1.05). This is done by recording a video of a static scene while slowly changing the camera's exposure. If only a small number of images or exposure times is available, a regularized approach will be required.

Non-parametric Vignette Calibration

We estimate a non-parametric (dense) vignetting map $V : \Omega \to [0, 1]$ from a sequence of images showing a planar scene. Apart from planarity, we only require the scene to have a bright (potentially non-uniform) color and to be fully Lambertian – in practice, a predominantly white wall serves well. This is in contrast to [99], which assumes a uniformly coloured flat surface. For simplicity, we estimate the camera pose with respect to the planar surface $\mathcal{P} \subset \mathbb{R}^3$ using an AR Marker [44]; however any other method, including a full monocular SLAM system can be used. For each image I_i, this results in a mapping $\pi : \mathcal{P} \to \Omega$ that projects a point on the 3D plane to a pixel in the image (if it is visible).

Again, we assume Gaussian white noise on $U(I_i(\pi_i(\mathbf{x})))$, leading to the Maximum-Likelihood energy

$$E(C, V) = \sum_{i, \mathbf{x} \in \mathcal{P}} \left(t_i V\Big(\big[\pi_i(\mathbf{x})\big]\Big) C(\mathbf{x}) - U\Big(I_i\big(\pi_i(\mathbf{x})\big)\Big)\right)^2, \tag{9.9}$$

where $C : \mathcal{P} \to \mathbb{R}$ is the unknown irradiance of the planar surface. In practice we define the surface to be square, and discretise it into 1000×1000 points; $[\cdot]$ then denotes rounding to the closest discretised position.

The energy $E(C, V)$ can be minimized alternatingly, again fixing one variable

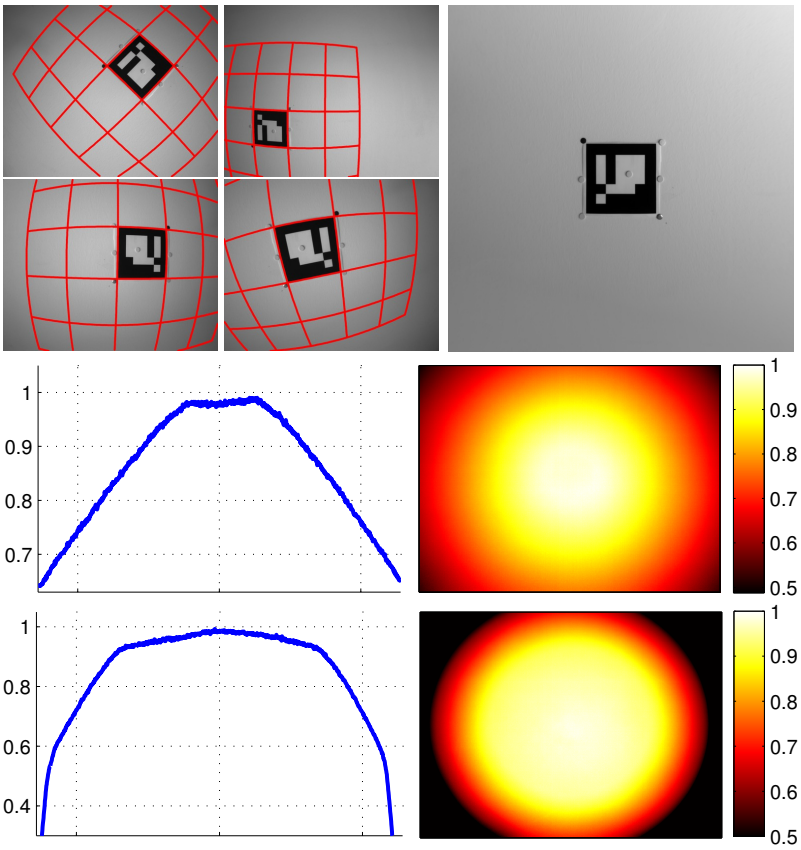


Figure 9.4: **Vignette calibration.** Top-left: four out of over 700 images used for vignette calibration, overlaid with the 3D plane $\mathcal{P}$ in red. Top-right: estimated irradiance image C for plane $\mathcal{P}$. Bottom-right: estimated dense attenuation factors V, for both used lenses. Bottom-left: horizontal cross-section through V at the middle of the image.

decouples all other unknowns, such that minimization becomes trivial:

$$\begin{aligned} C^*(\mathbf{x}) &= \underset{C(\mathbf{x})}{\operatorname{argmin}}\, E(C,V) \\ &= \frac{\sum_i t_i V\Big([\pi_i(\mathbf{x})]\Big) U\Big(I_i(\pi_i(\mathbf{x}))\Big)}{\sum_i \Big(t_i V([\pi_i(\mathbf{x})])\Big)^2}, \end{aligned} \tag{9.10}$$

$$\begin{aligned} V^*(\mathbf{x}) &= \underset{V(\mathbf{x})}{\operatorname{argmin}}\, E(C,V) \\ &= \frac{\sum_i t_i C(\mathbf{x}) U\Big(I_i(\pi_i(\mathbf{x}))\Big)}{\sum_i \Big(t_i C(\mathbf{x})\Big)^2}. \end{aligned} \tag{9.11}$$

Again, we do not impose any explicit smoothness prior or enforce certain properties (like radial symmetry) by finding a parametric representation for V. Instead, we choose to solve the problem by using large amounts (several hundred images) of data. Since V is only observable up to a scalar factor, we scale the result such that $\max(V) = 1$. Figure 9.4 shows the estimated attenuation factor map for both lenses. The high degree of radial symmetry and smoothness comes solely from the data term, without additional prior.

Note on Observability. Without regularizer, the optimization problem (9.9) is well-constrained if and only if the corresponding bipartite graph between all optimization variables is fully connected. In practice, the probability that this is not the case is negligible when using sufficient input images: Let (A, B, E) be a random bipartite graph with $|A| = |B| = n$ nodes and $|E| = [n(\log n + c)]$ edges, where c is a positive real number. We argue that the complex nature of 3D projection and perspective warping justifies the approximation of the resulting residual graph as random, provided the input images cover a wide range of viewpoints. Using the Erdős-Rényi theorem [39], it can be shown that for $n \to \infty$, the probability of the graph being connected is given by $P = e^{-2e^{-c}}$ [91]. In our case, $n \approx 1000^2$, which is sufficiently large for this approximation to be good. This implies that $30n$ residuals (i.e., 30 images with the full plane visible) suffice for the problem to be almost certainly well-defined ($P > 0.999999$). To obtain a good solution and fast convergence, a significantly larger number of input images is desirable (we use several hundred images) which are easily captured by taking a short (60s) video, that covers different image regions with different plane regions.

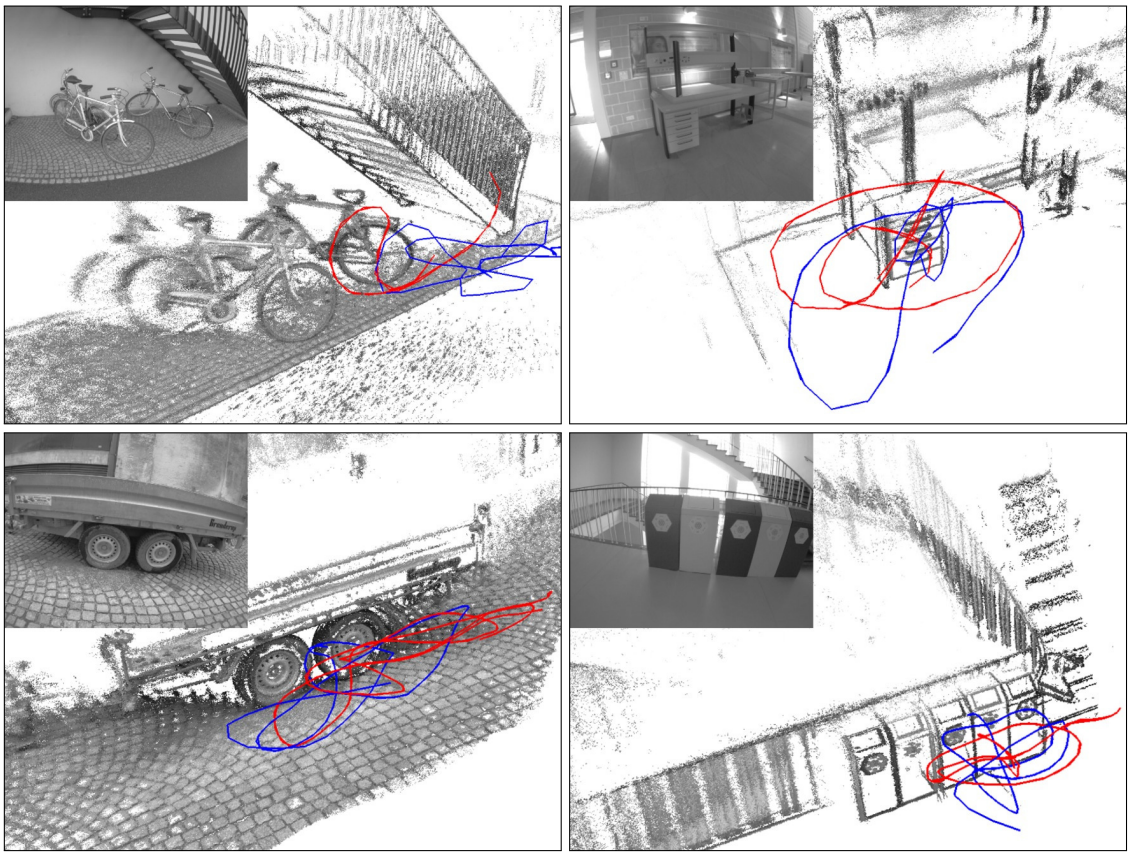


Figure 9.5: **Loop-closure alignment.** Explicit loop-closure alignment for 4 selected sequences, created with LSD-SLAM. The red and blue line correspond to the first and last segment of the full trajectory.

9.3 Evaluation Metrics

9.3.1 Evaluation from Loop-Closure

The dataset focuses on a large variety of real-world indoor and outdoor scenes, for which it is very difficult to obtain metric ground truth poses. Instead, all sequences contain exploring motion, and have one large loop-closure at the end: The first and the last 10-20 seconds of each sequence show the same, easy-to-track scene, with slow, loopy camera motion. We use LSD-SLAM [4] to track only these segments, and – very precisely – align the start and end segment, generating a "ground truth" for their relative pose[1]. To provide better comparability between the sequences, the ground truth scale is normalized such that the full trajectory has a length of approximately 100.

The tracking accuracy of a VO method can then be evaluated in terms of the accumulated error (drift) over the full sequence. **Note that this evaluation method is only valid if the VO/SLAM method does not perform loop-closure itself. To evaluate full SLAM systems like ORB-SLAM or LSD-SLAM, loop-closure detection needs to be disabled**. We argue that even for full SLAM methods, the amount of drift accumulated before closing the loop is a good indicator for the accuracy. In particular, it is strongly correlated with the long-term accuracy after loop-closure.

It is important to mention that apart from accuracy, full SLAM includes a number of additional important challenges such as loop-closure detection and subsequent map correction, re-localization, and long-term map maintenance (life-long mapping) – all of which are not evaluated with the proposed set-up.

9.3.2 Error Metric

Evaluation proceeds as follows: Let $p_1 \dots p_n \in \mathbb{R}^3$ denote the tracked positions of frames 1 to n. Let $S \subset [1; n]$ and $E \subset [1; n]$ be the frame-indices for the start- and end-segments for which aligned ground truth positions $\hat{p} \in \mathbb{R}^3$ are provided. First, we align the tracked trajectory with both the start- and end-segment independently, providing two relative transformations

$$T_s^{\text{gt}} := \underset{T \in \text{Sim}(3)}{\text{argmin}} \sum_{i \in S} (T p_i - \hat{p}_i)^2 \tag{9.12}$$

$$T_e^{\text{gt}} := \underset{T \in \text{Sim}(3)}{\text{argmin}} \sum_{i \in E} (T p_i - \hat{p}_i)^2. \tag{9.13}$$

For this step it is important, that both E and S contain sufficient poses in a non-degenerate configuration to well-constrain the alignment – hence the loopy motion

[1] Some sequences begin and end in a room which is equipped with a motion capture system. For those sequences, we use the metric ground truth from the MoCap.

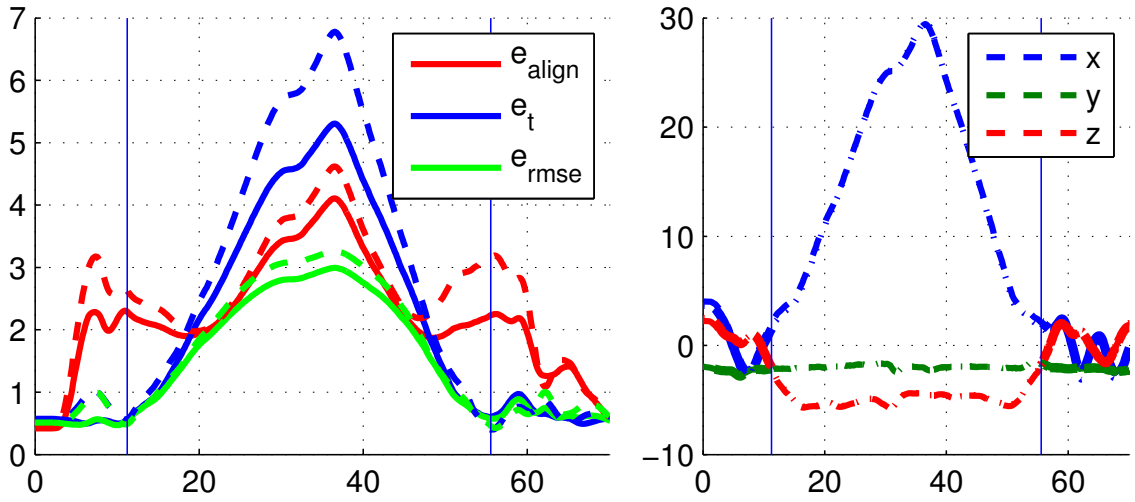


Figure 9.6: **Evaluation metric.** The right plot shows a tracked (x, y, z)-trajectory for `sequence_16`. We analyse the effect of drift on the different error metrics by adding an artificial scale-jump of $\times 0.8$ or rotation-jump of $10°$ at different time-points throughout the sequence: The left plot shows how the error metrics depend on the time-point where drift occurs (scale: dashed, rotation: solid). Both e_t and e_{rmse} are heavily correlated with *where* the drift occurs (the further away, the more impact it has). The alignment error e_{align} in contrast behaves much more stable, i.e., the error is less susceptible to where the drift occurs.

patterns at the beginning and end of each sequence. The accumulated drift can now be computed as $T_{\text{drift}} = T_e^{\text{gt}}(T_s^{\text{gt}})^{-1} \in \text{Sim}(3)$, from which we can explicitly compute (a) the scale-drift $e_s := \text{scale}(T_{\text{drift}})$, (b) the rotation-drift $e_r := \text{rotation}(T_{\text{drift}})$ and (c) the translation-drift $e_t := \|\text{translation}(T_{\text{drift}})\|$.

We further define a combined error measure, the *alignment error*, which equally takes into account the error caused by scale, rotation and translation drift over the full trajectory as

$$e_{\text{align}} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|T_s^{\text{gt}} p_i - T_e^{\text{gt}} p_i\|_2^2}, \tag{9.14}$$

which is the translational RMSE between the tracked trajectory, when aligned (a) to the start segment and (b) to the end segment. Figure 9.7 shows an example. We choose this metric, since

- it can be applied to other SLAM / VO modes with different observability modes (like visual-inertial or stereo),
- it is equally affected by scale, rotation, and translation drift, implicitly weighted by their effect on the tracked position,

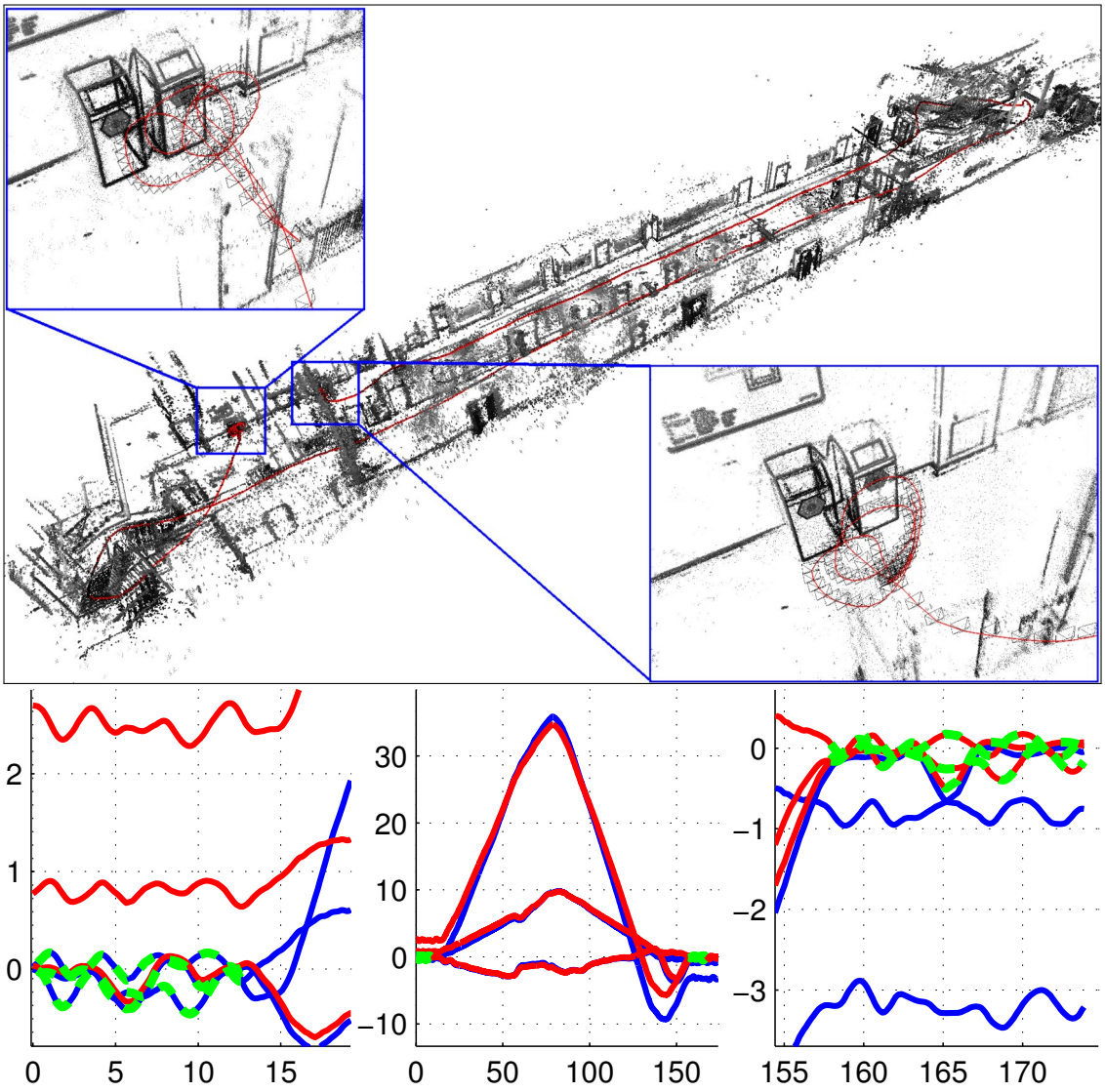


Figure 9.7: **Alignment error.** The top shows `sequence_40`, tracked using [3]. The accumulated drift is clearly visible in the reconstruction (the two enlarged segments should overlap). The bottom plots show the loop-closure ground truth (dashed, green), and the tracked trajectory (1) aligned to the start-segment in blue and (2) aligned to the end segment in red (the center plot shows the full trajectory, the left and right plot show a close-up of the start- and end-segment respectively). The alignment error e_{align} computes the RMSE between the red and the blue line, over the full trajectory. For this example, $e_{\text{align}} = 2.27$, $e_{\text{s}} = 1.12$ and $e_{\text{r}} = 3.9°$.

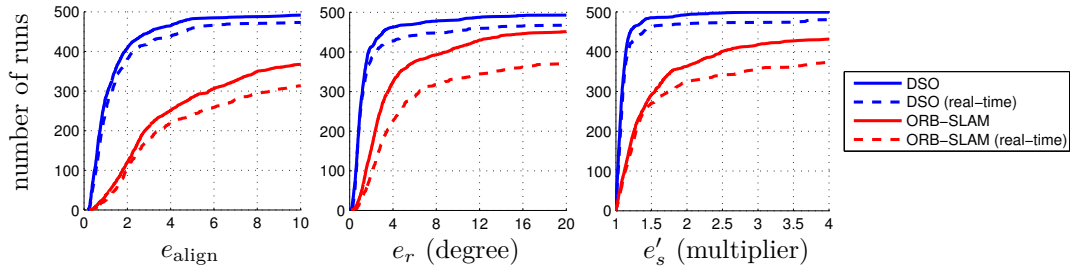


Figure 9.8: **Evaluation result.** Cumulative error-plots over all 50 sequences, run forwards and backwards, 5 times each to account for non-deterministic behaviour. For each error-value (x-axis), the plot shows the number of runs (y-axis) for which the achieved error was smaller. Note that since e_s is a multiplicative factor, we summarize $e'_s = \max(e_s, e_s^{-1})$. The solid line corresponds to non-real time execution, the dashed line to hard enforced real-time processing.

- it can be applied for algorithms which compute only poses for a subset of frames (e.g. keyframes), as long as start- and end-segment contain sufficient frames for alignment, and
- it better reflects the overall accuracy of the algorithm than the translational drift drift e_t or the joint RMSE

$$e_{\text{rmse}} := \sqrt{\min_{T \in \text{Sim}(3)} \frac{1}{|S \cup E|} \sum_{i \in S \cup E} (T p_i - \hat{p}_i)^2}, \tag{9.15}$$

  as shown in Figure 9.6. In particular, e_{rmse} becomes degenerate for sequences where the accumulated translational drift surpasses the standard deviation of $\hat{p}$, since the alignment (9.15) will simply optimize to $\text{scale}(T) \approx 0$.

9.4 Benchmark

When evaluating accuracy of SLAM or VO methods, a common issue is that not all methods work on all sequences. This is particularely true for monocular methods, as sequences with degenerate (rotation-only) motion or entirely texture-less scenes (white walls) cannot be tracked. All methods will then either produce arbitrarily bad (random) estimates or heuristically decide they are "lost", and not provide an estimate at all. In both cases, averaging over a set of results containing such outliers is not meaningful, since the average will mostly reflect the (arbitrarily bad) errors when tracking fails, or the threshold when the algorithm decides to not provide an estimate at all.

A common approach hence is to show only results on a hand-picked subset of sequences on which the compared methods do not fail (encouraging manual overfitting), or to show large tables with error values, which is not practicable for a dataset

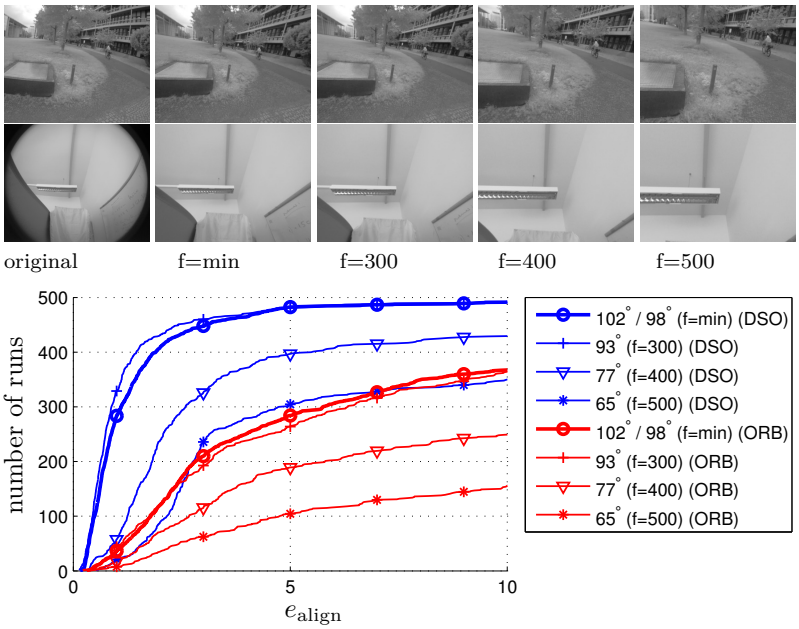


Figure 9.9: **Different field of view.** Alignment error when changing the horizontal field of view (i.e., using different focal lengths f for the rectified images). The top shows two example images, rectified with the different focal lengths. "f=min" refers to the default setting, which differs for the two used lenses – as comparison, the horizontal field of view of a Kinect camera is 63°. A smaller field of view significantly decreases accuracy and robustness, for both methods (note that for a cropped field of view, some sequences contain segments with only a white wall visible).

containing 50 sequences. A better approach is to summarize tracking accuracy as cumulative distribution, visualizing on how many sequences the error is below a certain threshold – it shows both the accuracy on sequences where a method works well, as well as the method's robustness, i.e., on how many sequences it does not fail.

Figure 9.8 shows such cumulative error-plots for two methods, DSO (Direct Sparse Odometry) [3] and ORB-SLAM [86], evaluated on the presented dataset. Each of the 50 sequences is run 5 times forwards and 5 times backwards, giving a total of 500 runs for each line shown in the plots.

Algorithm Parameter Settings. Since both methods do not support the FOV camera model, we run the evaluation on pinhole-rectified images with VGA (640×480) resolution. Further, we disable explicit loop-closure detection and re-localization to allow application of our metric, and reduce the threshold where ORB-SLAM decides it is lost to 10 inlier observations. Note that we do not impose any restriction on implicit, "small" loop-closures, as long as these are found by ORB-SLAM's local mapping component (i.e., are included in the co-visibility graph).

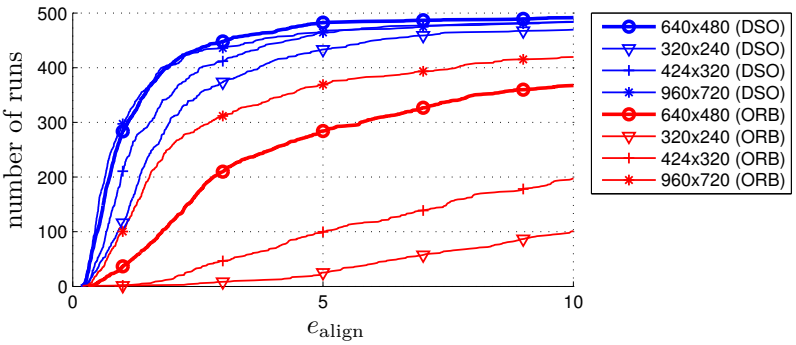


Figure 9.10: **Different image resolution.** Alignment error when changing the rectified image resolution. Note that the algorithms were not run in real time, hence the change in computational complexity is not accounted for. While for ORB-SLAM the resolution has a strong effect, DSO is only marginally affected – which is due to the sub-pixel-accurate nature of direct approaches.

Since DSO does not perform loop-closure or re-localization, we can use the default settings. We run both algorithms in a non-real-time setting (at roughly one quarter speed), allowing to use 20 dedicated workstations with different CPUs to obtain the results presented in this paper. Figure 9.8 additionally shows results obtained when hard-enforcing real-time execution (dashed lines), obtained on the same workstation which is equipped with an i7-4910MQ CPU.

Data Variations. A good way to further to analyse the performance of an algorithm is to vary the sequences in a number of ways, simulating different real-world scenarios:

- Figure 9.9 shows the tracking accuracy when rectifying the images to different fields of view, while keeping the same resolution (640×480). Since the raw data has a resolution of 1280×1024, the caused distortion is negligible.
- Figure 9.10 shows the tracking accuracy when rectifying the images to different resolutions, while keeping the same field of view.
- Figure 9.11 shows the tracking accuracy when playing sequences only *forwards* compared to the results obtained when playing them only *backwards* – switching between predominantly forward-motion and predominantly backward-motion.

In each of the three figures, the bold lines correspond to the default parameter settings, which are the same across all evaluations.

Ground Truth Validation. Since for most sequences, the used loop-closure ground truth is computed with a SLAM-algorithm (LSD-SLAM) itself, it is not

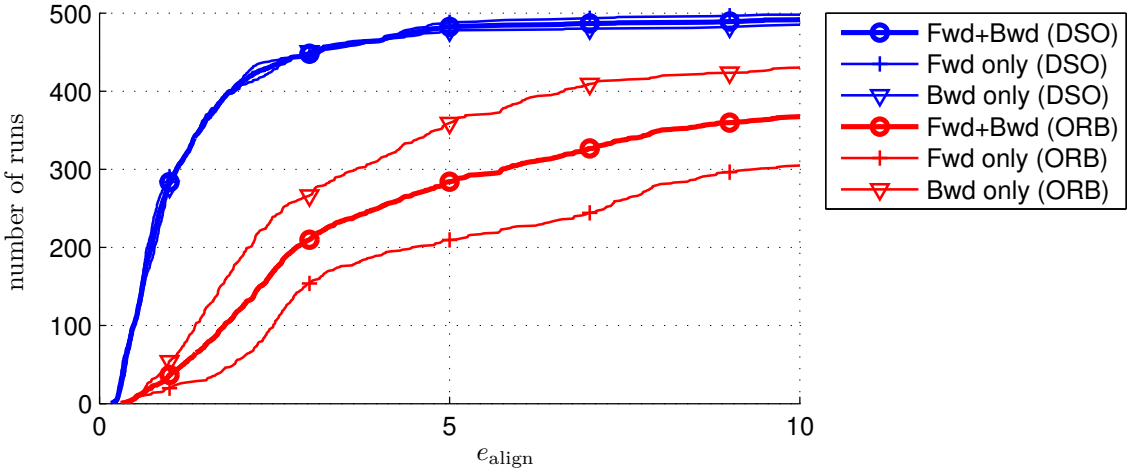


Figure 9.11: **Dataset motion bias.** Alignment error when running all sequences forwards and backwards, as well as the combination of both (default): While DSO is largely unaffected by this, ORB-SLAM performs significantly better for backwards-motion. This is a classical example of "dataset bias", and shows the importance of evaluating on large datasets, covering a diverse range of environments and motion patterns.

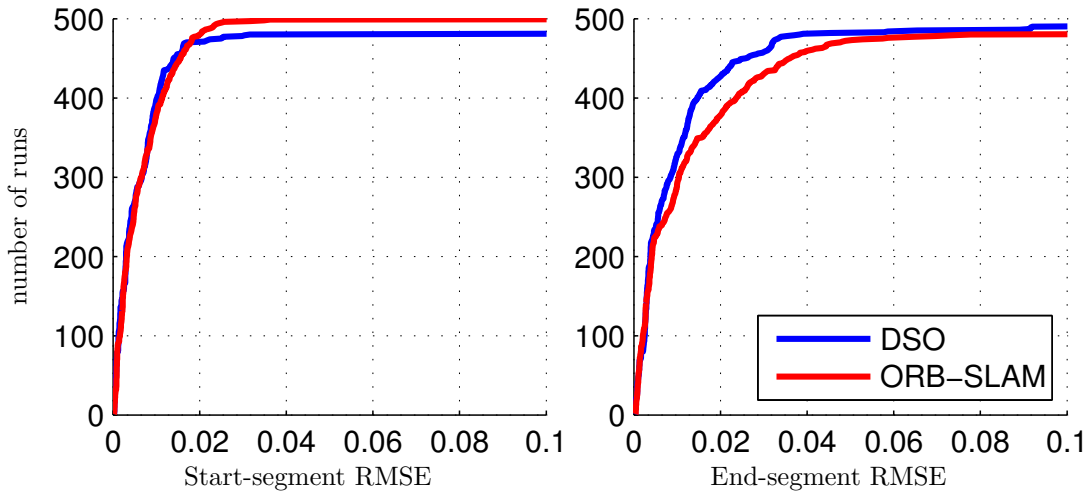


Figure 9.12: **Start- and end-segment error.** RMSE for alignment with the provided start- and end-segment ground truth. Note that it is roughly 100 times lower than e_{align}, and very similar for both evaluated methods. It can also be observed, that the automatic initialization of DSO fails occasionally (in that case all errors, including the start-segment error, are set to infinity), while this is not the case for ORB-SLAM. If the algorithm fails to provide an estimate for the full trajectory, the end-segment error is set to infinity.

perfectly accurate. We can however validate it by looking at the RMSE when aligning the start- and end-segment, i.e., the minima of (9.12) and (9.13) respectively. They are summarized in Figure 9.12. Note the difference in order of magnitude: The RMSE within start- and end-segment is roughly 100 times smaller than the alignment RMSE, and very similar for both evaluated methods. This is a strong indicator that almost all of the alignment error originates from accumulated drift, and not from noise in the ground truth.

9.4.1 Dataset

The full dataset, as well as preview-videos for all sequences are available on

`http://vision.in.tum.de/mono-dataset`

We provide

- Raw camera images of all 50 sequences (43GB; 190'000 frames in total), with frame-wise exposure times and computed ground truth alignment of start- and end-segment.
- Geometric (FOV distortion model) and photometric calibrations (vignetting and response function).
- Calibration datasets (13GB) containing (1) checkerboard-images for geometric calibration, (2) several sequences suitable for the proposed vignette and response function calibration, (3) images of a uniformly lit white paper.
- Minimal c++ code for reading, pinhole-rectifying, and photometrically undistorting the images, as well as for performing photometric calibration as proposed in Section 9.2.3.
- Matlab scripts to compute the proposed error metrics, as well as the raw tracking data for all runs used to create the plots and figures in Section 9.4.

9.4.2 Known Issues

- Even though we use industry-grade cameras, the SDK provided by the manufacturer only allows to asynchronously query the current exposure time. Thus, in some rare cases, the logged exposure time may be shifted by one frame.

Part III

Conclusion & Outlook

Chapter 10

Contribution

We have developed **direct** approaches for one of the most fundamental tasks in computer vision, estimating 3D geometry and camera motion from 2D images. While direct formulations have a long history in the literature – first direct methods even pre-dating the advent of keypoint descriptors and detectors – the state of the art in real-time localization and mapping (SLAM) and visual odometry (VO) predominantly operates on small sets of extracted keypoint matches.

10.1 Thesis Summary

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

Chapter 3: Semi-Dense Visual Odometry for a Monocular Camera. We proposed a frame-to-frame, real-time direct visual odometry approach. It introduced the concept of incrementally estimating semi-dense depth maps in a probabilistic framework, using pixel-wise filtering over many small-baseline photometric stereo comparisons. Furthermore, we introduced the concept of *geometric* vs. *photometric* noise in the input images, and derive a model of how they affect photometric, stereo-based depth estimation accuracy. The distinction between photometric and geometric noise is picked up again in Chapter 8, where we analyzed the different effects of these two types of noise on the direct and the indirect model.

Chapter 4: LSD-SLAM: Large-Scale Direct Monocular SLAM. We extend the previously presented semi-dense odometry formulation to a full SLAM system, including loop-closure detection and global map optimization. To this end, we introduced keyframes – semi-dense depth maps with associated Gaussian depth uncertainties – and keyframe-to-keyframe alignment on Sim(3), explicitly estimating accumulated scale-drift. This allows efficient global map optimization and loop-

closure correction in a pose-graph framework. The inclusion of scale in the optimization is required due to the separation between pose- and geometry estimation, fixing the estimated local geometry (and thus the scale of a keyframe) to a – estimated, hence potentially erroneous – scale. The resulting real-time monocular SLAM system creates high-fidelity semi-dense maps of the environment in real-time on a CPU, and has been published as open-source code.

Chapter 5: Semi-Dense Visual Odometry for AR on a Smartphone. We developed an Android implementation of the previously presented direct semi-dense odometry approach, demonstrating that the computational demands of direct formulations are not necessarily higher than those of comparable indirect methods. Furthermore, we showed integration into an augmented reality framework, including basic physical interaction with the environment (driving a car on a flat surface).

Chapter 6: Large-Scale Direct SLAM with Stereo Cameras. We proposed a generalization of LSD-SLAM to stereo cameras. It combines information from temporal stereo (disparity information from images taken by the same camera at different points in time) and static stereo (disparity information from images taken by different cameras, at the same point in time). In addition, invariance to automatic exposure changes is added by including an affine intensity transfer function between frames. The resulting SLAM system achieves state-of-the-art accuracy and runtime on challenging benchmarks such as KITTI and the EuRoC MAV sequences, while at the same time reconstructing high-fidelity semi-dense maps of the environment. Furthermore, we analyzed the evolution of runtime and tracking accuracy with changing image resolution – exploiting the sub-pixel accurate nature of direct methods. This demonstrates the ability to generalize to very low image resolutions, while still achieving competitive tracking accuracy, while using only minimal computational resources.

Chapter 7: Large-Scale Direct SLAM for Omnidirectional Cameras. We showed a generalization of LSD-SLAM to cameras with a field of view above 180°. In particular, we generalized direct image alignment and incremental depth estimation to two camera models which allow to represent a field of view above 180°, the unified omnidirectional model as well as a cube-shaped piecewise pinhole model. Furthermore, we transitioned from representing inverse depth to inverse distance, avoiding the singularity at $z = 0$.

Chapter 8: Direct Sparse Odometry. With DSO, we introduced an alternative, sparse and direct direct visual odometry formulation. It retains the geometry representation (inverse depth) and the direct energy formulation from LSD-SLAM. In contrast to LSD-SLAM however, geometry and camera poses are optimized in a

joint Gauss-Newton framework, preserving statistical consistency and avoiding premature convergence as well as the accumulation of linearization errors. To facilitate this in real-time, we drop the smoothness prior employed by LSD-SLAM, switching to a sparse geometry representation – nonetheless, our approach retrains the ability to sample from across all information including edges, instead of only using corners. Furthermore, we include the camera intrinsics in the optimization, and add complete photometric calibration to the model – including exposure time, non-linear pixel response and lens vignetting. We analyze the effect of a number of fundamental design and parameter choices using a large dataset of real-world sequences.

Chapter 9: A Photometrically Calibrated Benchmark For Monocular Visual Odometry. This publication is closely related to DSO, and describes the dataset on which many of the parameter- and tracking accuracy evaluations for DSO are performed. It extends on existing datasets in a number of ways: First, it contains a complete photometric calibration, including frame-wise exposure times, the calibrated pixel response function and lens vignetting factors, for which we present novel calibration approaches which require minimal set-up. Second, it surpasses existing public datasets in scale and variety, containing 50 sequences recorded in indoor and outdoor environments, featuring different motion patterns, two different lenses, a variety of frame-rates, and different sensor settings. In total, the dataset contains 105 minutes of video (190'000 frames). To facilitate the scale of the benchmark, we propose a novel tracking accuracy evaluation metric – the *alignment error*. It is based on the accumulated drift after a large loop – equally weighting scale-, rotation- and translation drift based on their influence on position estimation error – thereby not requiring ground-truth poses for every frame in the sequence.

10.2 Gained Experimental Insights

Using the TUM monoVO dataset and evaluation methodology presented in the last chapter, we performed extensive parameter studies, thoroughly evaluating the effect of algorithm and system design choices. In addition, we analyzed the effect of different types of noise in the images on the direct and the indirect formulation. In total, the parameter and accuracy evaluations from Chapter 8 and 9 summarize 40'000 tracked sequences (100 million tracked frames, or 38 days of video at 30fps), computed on a cluster of 20 workstations. Our findings include:

- **Field of view:** A large field of view (at fixed video resolution) increases tracking accuracy and robustness for both direct and indirect methods, even though this implies decreased angular image resolution.
- **Video resolution:** The video resolution (at fixed field of view) has a significantly larger effect on indirect approaches than on direct approaches. This

is consistent with the results from Chapter 6, showing that direct approaches are better at extracting information from low-resolution images, which may only contain few detectable keypoints. It also supports the observation that the accuracy of DSO is not limited by the amount of available data, but rather by the limited expressive complexity of the underlying predictive model. In turn, indirect approaches require high image resolution and more feature-rich environments to reach this threshold, and are in fact limited by the amount of (usable) data in practice.

- **Dataset bias:** We demonstrate the importance of evaluating on large, diverse datasets, by analyzing the tracking accuracy when running the same sequences forwards and backwards. While DSO's tracking accuracy is the same in both cases, the accumulated drift of ORB-SLAM is roughly twice as large for forward-motion as for backward-motion.

- **Image data redundancy:** We show that in practice, image data is locally highly redundant, and that there is little to no benefit of using more than a certain number of data points (800 per keyframe in our experiments) in terms of tracking accuracy. This is likely due to other error sources (i.e., systematic biases arising from various model violations, such as reflections, geometric aliasing, occlusions, or deviations of the real-world lens projection from the used parametric, central approximation) becoming the dominant error source. Note that we have only analyzed this in terms of long-term pose estimation accuracy – where, even with heavily sub-sampled data and after eliminating / marginalizing 3D geometry parameters, there are several orders of magnitude more constraints than degrees of freedom. The amount of data required to reconstruct accurate and complete 3D models is much larger, rendering the direct approaches ability to extract more information from available image data more valuable.

- **Additional value of non-corners:** While adding more data does not necessarily increase accuracy and robustness, the ability of direct methods to sample data points from across all available information – including edges and weakly textured surfaces – does increase tracking accuracy and robustness.

- **Impact of prematurely fixing linearizations:** Our experiments indicate that – in the DSO implementation – taking keyframes more frequently decreases tracking accuracy. This is due to the fixed windows size: taking more keyframes causes each keyframe to be marginalized earlier – thereby fixing linearizations more early on, and around a linearization point further away from the correct estimate. Since LSD-SLAM fixes linearizations immediately (at the very first evaluation point for depth estimation), DSO's ability to relinearise as better state estimates become available is likely to be one of the major sources of increased tracking accuracy compared to LSD-SLAM.

- **Photometric vs. geometric noise:** We show that the indirect formulation is much more robust to geometric noise in the data, for example caused by a rolling shutter or inaccurate intrinsic calibrations. In turn, the direct formulation is significantly more accurate and robust on geometrically well-calibrated data, and more robust to photometric noise (such as motion blur).

Chapter 11

Limitations and Future Research

All methods and models developed in this thesis are entirely based on a direct model formulation and do not rely on corner detectors or keypoint descriptors. While our evaluations on extensive datasets have shown the benefits of a direct formulation in terms of tracking accuracy and robustness, we have also discussed the shortcomings of a direct approach. In this chapter we first discuss the limitations of a direct structure and motion formulation, and subsequently provide a summary of promising future research directions.

11.1 Limitations

First, the most fundamental limitation of direct methods lies in the high degree of non-linearity of the photometric error terms, causing the least-square energy function to be strongly non-convex. This limits direct approaches to incremental tracking, as they require accurate initializations for all involved parameters. In fact, DSO spends half of the compute budget – and more than half of the implementation's lines of code – solely on providing *initializations* for the back-end minimization, and failure to provide these is the main sources of tracking failure. For camera pose parameters, the need for initializations can be removed by tightly integrating an IMU as done in [12]. Geometry parameters on the other are initialized using a discrete search along the epipolar line both in LSD-SLAM as well as DSO, which only works well for small inter-frame motion. This currently restricts direct formulations to video processing, rather than 3D reconstruction from unordered sets of photographs. Furthermore, keypoints enable efficient approaches for loop-closure detection and relocalization that scale logarithmically with the map size – while the direct loop-closure detection approach implemented in this thesis (as, e.g., described in Chapter 6) relies on (guided) sampling, which has linear worst-case complexity, and thus will not scale well to arbitrarily large scenes.

Second, we have shown that the performance of the direct model quickly

deteriorates in the presence of geometric noise. This is because (1) geometric noise is not modeled, and (2) in the presence of geometric noise larger than 2 pixels, there likely exists no solution / initialization for which all residuals are within the validity range of the image linearization, thus optimization is likely to fail entirely – in this case, reverting to a coarser image resolution will give better results. In practice, this means that direct approaches require a more accurate (geometric) sensor model than indirect formulations, and cannot afford to ignore rolling shutter or radial lens distortions.

Third, the inclusion of zeroth-order information (absolute intensity) makes the direct approach susceptible to illumination changes. While we have shown that auto-exposure and global, affine illumination changes can be compensated for by including them in the model, local, non-affine changes (e.g., caused by moving light sources) severely impact the performance of the direct formulation. This can be alleviated by switching to a first-order or second-order error formulation, i.e., formulating residual terms on first or second order image derivatives. Even invariance to arbitrary monotonic brightness changes – as offered by modern binary keypoint descriptors like ORB – can be achieved by formulating error residuals on the local image gradient direction.

11.2 Future Research

There is a number of interesting future research directions, which we sketch out in the following. We start with clearly defined and immediate extensions, and lead up to more long-term research goals.

Practical extensions to DSO. We have shown for LSD-SLAM how to (1) formulate it for omnidirectional cameras, (2) extend it to stereo- or multi-camera set-ups, and (3) tightly integrate an IMU. For DSO however – which offers significantly better tracking accuracy due to the joint, consistent optimization – these extensions remain open challenges, and are likely to significantly enhance the methods performance for practical application.

Furthermore, DSO can be extended to a full SLAM system by replacing the currently marginalization-based back-end with more flexible alternatives such as a double-window formulation. In fact, since all geometry parameters are represented in camera coordinate frames (and thus residuals only depend on pose transformations between frames, and not on absolute poses), geometry parameters can be marginalized / conditioned on in a strictly relative formulation. This potentially allows to perform efficient global map optimization on a pose-graph-like representation, without the need to re-evaluate the photometric residuals in each iteration.

Non-parametric geometric camera calibration. One of the most interesting insights from the experiments conducted in the last two chapters is, that the tracking error accumulated by DSO cannot be further decreased by (1) using more data (more points / frames), or (2) using more accurate data (higher resolution images). In fact, this indicates that the accumulated drift is not due to sensor noise – instead, it likely is caused by (1) outliers the front-end fails to recognize as such (reflections, occlusions / perceptual aliasing, slowly moving objects such as clouds), or (2) systematic biases induced by implicit model assumptions. One particularly influential assumption is the camera projection model, or rather its parametric approximation: In addition to the zero-aperture approximation employed by all central camera models, structure and motion methods generally approximate the true projection function – mapping 3D points to 2D pixel coordinates – with a parametric expression; some examples were given in Chapter 2.1.1. This can be avoided by calibrating a non-parametric projection model, modelling the ray direction for each pixel independently, while using a parametric base-model as probabilistic prior, rather than as hard constraint.

Alternative error formulations. As mentioned in the previous section, one of the major drawbacks of the error formulation employed throughout this thesis is the incorporation of all zeroth-order information (absolute intensity), limiting the approach to scenes with constant illumination. This can be avoided – while staying in a direct framework – by employing error formulations defined on first- or second-order image derivatives, adding invariance to additive or affine illumination changes respectively. Invariance to arbitrary monotonic brightness changes can be obtained by defining an error metric on the local image gradient direction. It is important to note, that zeroth- and first-order information can provide valuable information that cannot be recovered otherwise, in particular on photometrically well-calibrated data. Examples include smooth intensity variations across mostly white walls.

Interesting research directions arise from combining multiple error metrics, for instance using zeroth-order residuals for a real-time visual-odometry front-end (where constant scene lighting is a reasonable assumption), while using higher-order residuals for loop-closures and to align images taken at very different points in time (where the scene lighting conditions may have changed significantly). Similarly, the best error metric can be chosen on a per-residual basis, attempting to maximize the retained information while gaining invariance to distortions present in the particular observation.

Densification and integration of semantic / learnt priors. The perhaps most exciting direction that will be taken in the future is the incorporation of learnt prior knowledge to regularize weakly observable, and to complete unobservable information, thereby paving the way to high-fidelity dense reconstruction from passive vision. We believe the acquisition and formulation of realistic, real-world priors on the 3D shape and appearance of the world we live in (replacing a basic smoothness

assumption) to be an essential requirement for this path.

A promising approach is to formulate this as hierarchical replacement scheme – incrementally marginalizing reoccurring geometric structures into a "vocabulary" of shapes learnt from real-world 3D scenes, starting at the level of simple geometric primitives such as planes, cylinders or rectangular corners, and leading all the way up to chairs, tables, cars or entire buildings. Such approaches will allow to complete unobservable information, while at the same time reducing the model's degrees of freedom (and thereby both required compute as well as memory), enabling truly large-scale, dense, and potentially non-rigid 3D reconstruction.

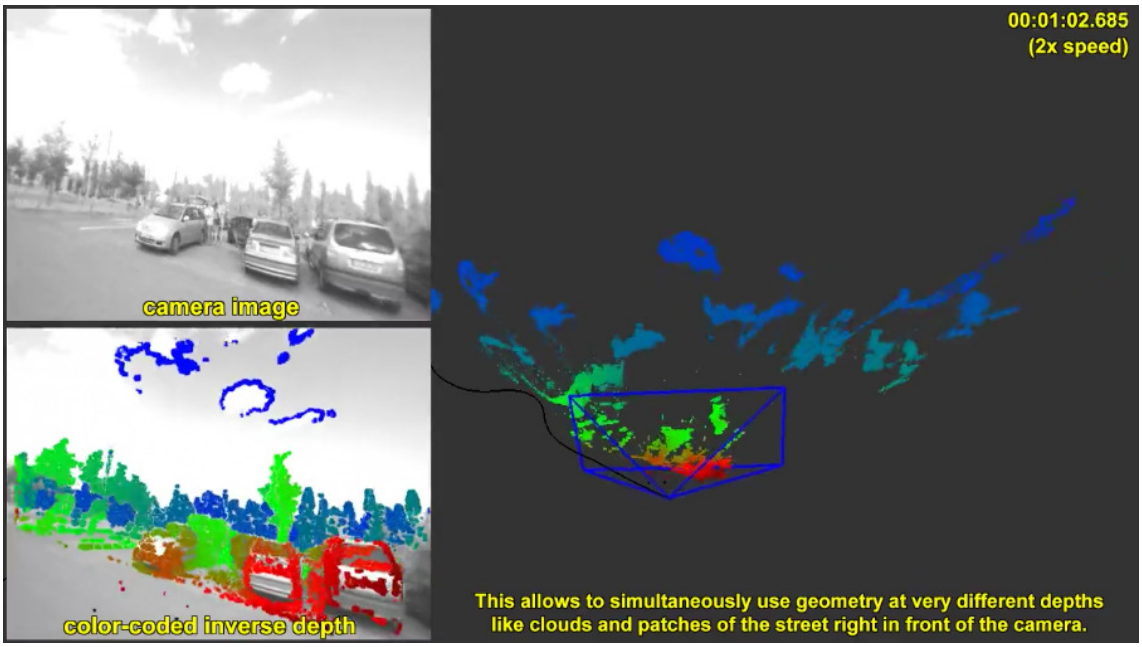
Part IV

Appendix

Appendix A

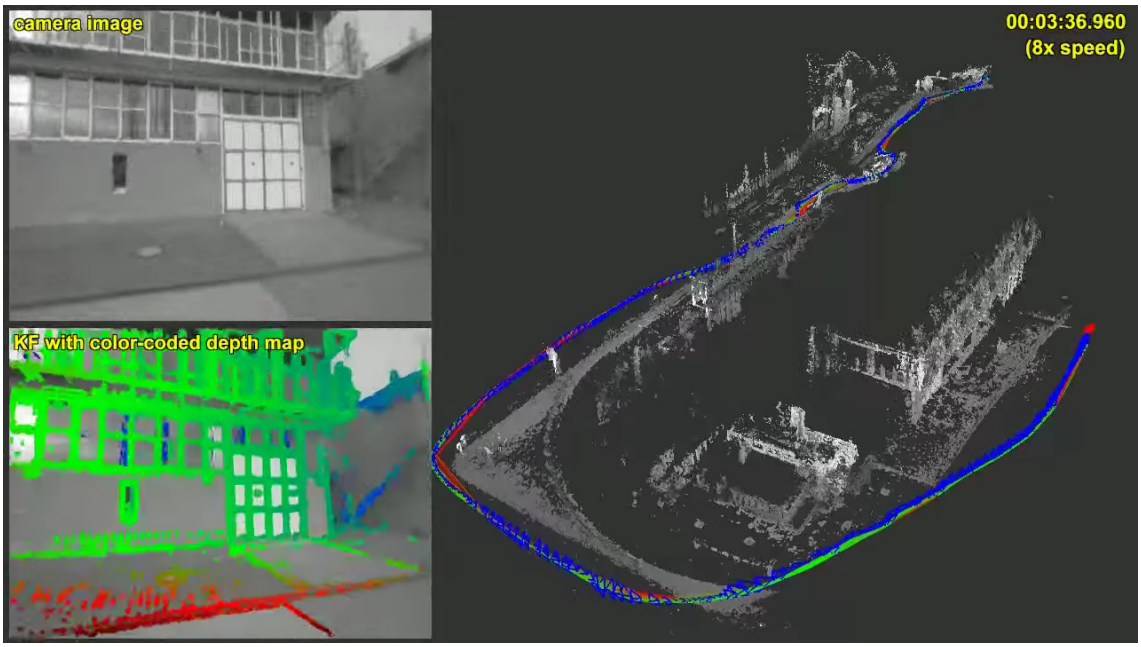
Multimedia Material

Semi-Dense Visual Odometry for a Monocular Camera



https://youtu.be/LZChzEcLNzI

LSD-SLAM: Large-Scale Direct Monocular SLAM

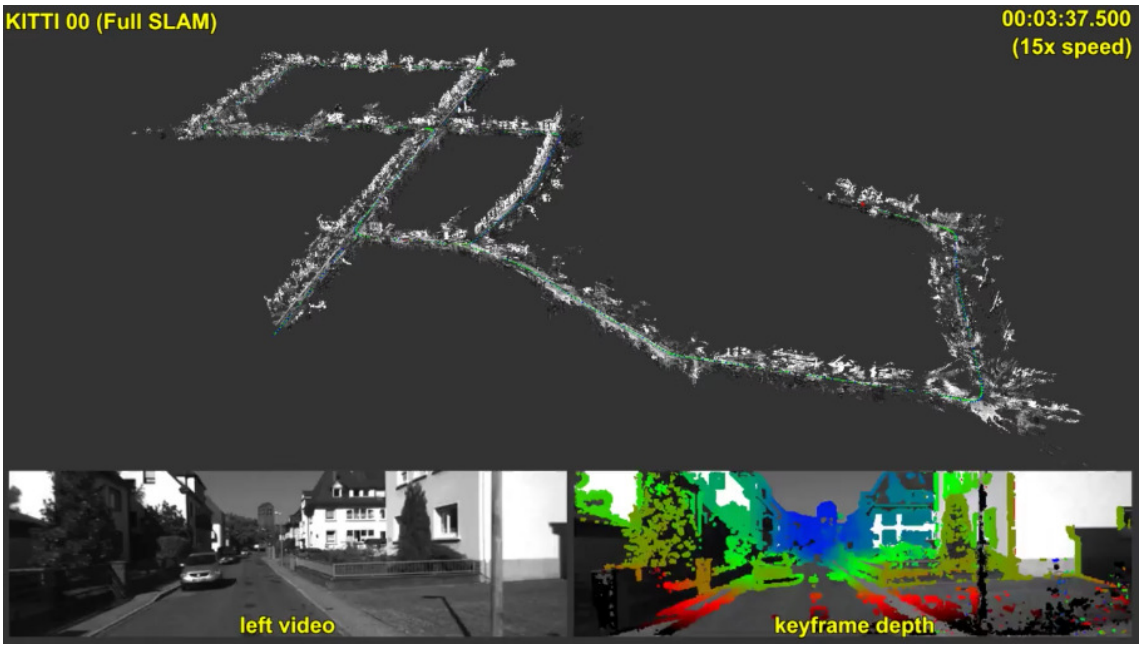


https://youtu.be/GnuQzP3gty4

Semi-Dense Visual Odometry for AR on a Smartphone

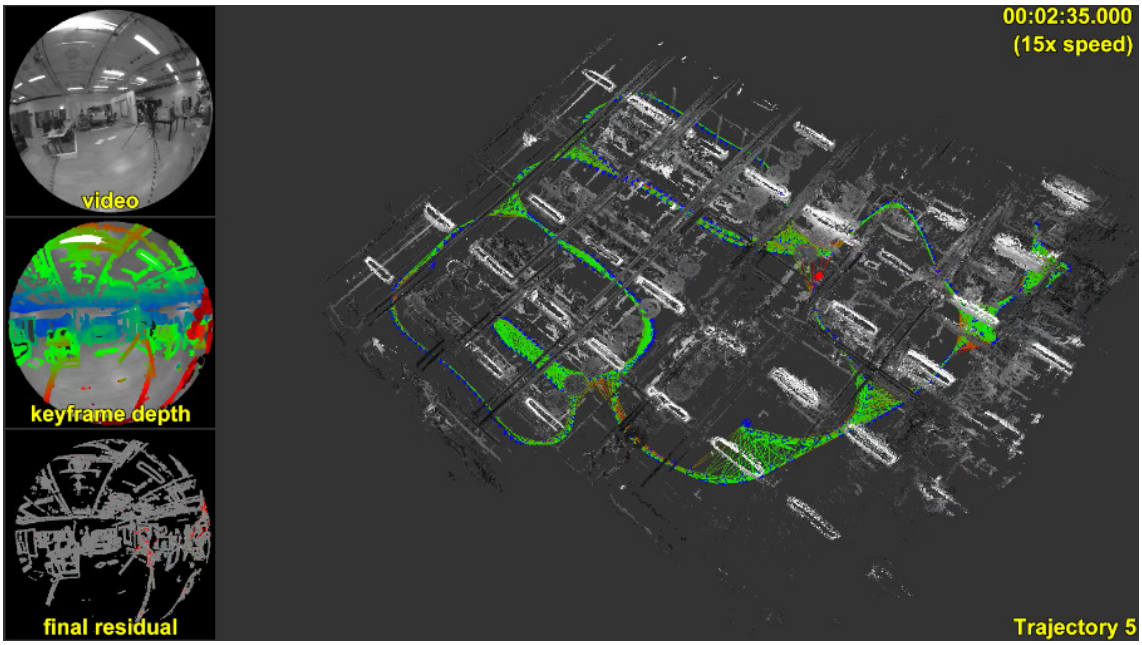
https://youtu.be/X0hx2vxxTMg

Large-Scale Direct SLAM with Stereo Cameras



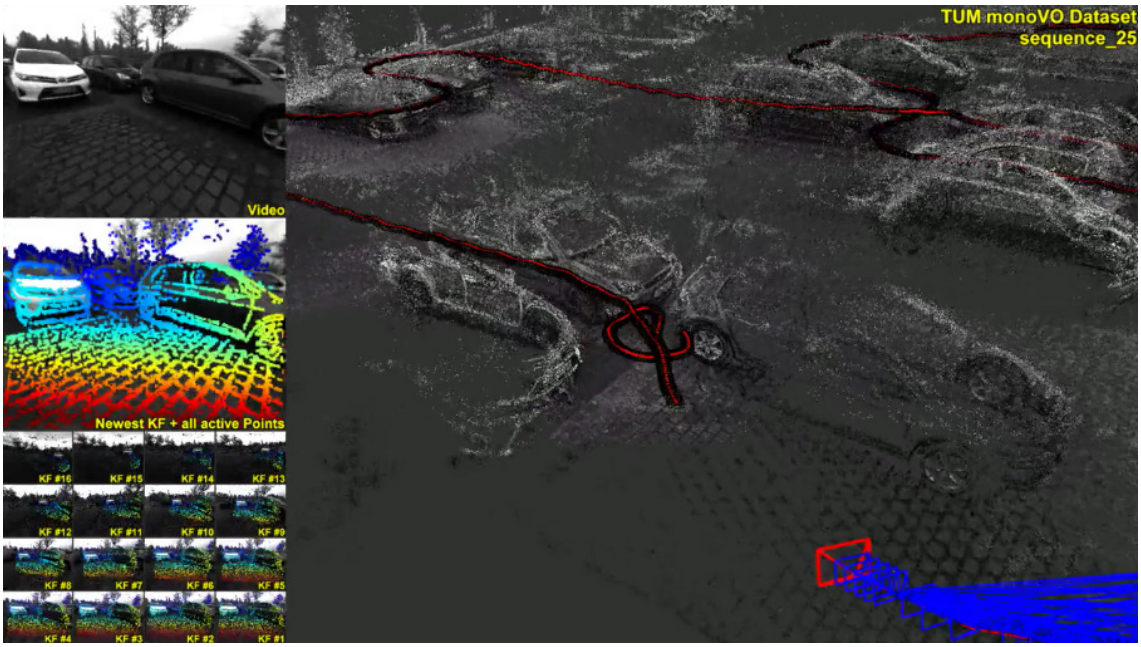
https://youtu.be/oJt3Ln8H03s

Large-Scale Direct SLAM for Omnidirectional Cameras



https://youtu.be/v0NqMm7Q6S8

DSO: Direct Sparse Odometry



https://youtu.be/C6-xwSOOdqQ

Appendix B

Open-Source Code and Datasets

LSD-SLAM: Large-Scale Direct Monocular SLAM

- `https://github.com/tum-vision/lsd_slam`: C++ implementation of LSD-SLAM (see Chapter 4).
- `https://vision.in.tum.de/lsdslam`: Four sequences used for the qualitative evaluation of LSD-SLAM (see Chapter 4).

Large-Scale Direct SLAM for Omnidirectional Cameras

- `https://vision.in.tum.de/omni-lsdslam`: Five sequences used for the quantitative evaluation of omnidirectional LSD-SLAM, with associated ground truth poses (see Chapter 7).

DSO: Direct Sparse Odometry

- `http://vision.in.tum.de/dso`: C++ implementation of DSO (see Chapter 8). *To appear.*

TUM monoVO Dataset

- `https://github.com/tum-vision/mono_dataset_code`: C++ / Matlab implementation for photometric camera calibration, and TUM monoVO dataset evaluation (see Chapter 9)
- `https://vision.in.tum.de/mono-dataset`: The full TUM monoVO dataset (see Chapter 9).

List of Figures

Own Publications

[1] D. Caruso, J. Engel, and D. Cremers. "Large-Scale Direct SLAM for Omnidirectional Cameras". In: *International Conference on Intelligent Robot Systems (IROS)*. 2015 (cited on pp. 11, 12, 14, 138).

[2] O. Dunkley, J. Engel, J. Sturm, and D. Cremers. "Visual-Inertial Navigation for a Camera-Equipped 25g Nano-Quadrotor". In: *Proceedings of the IROS Aerial Open Source Robotics Workshop*. 2014 (cited on pp. 9, 12).

[3] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: *arXiv:1607.02565*. 2016 (cited on pp. 11, 12, 14, 16, 164, 168, 177, 179).

[4] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. 2014 (cited on pp. 11, 12, 14, 78, 82, 89, 94, 95, 99, 101, 112–114, 119–123, 134, 151, 164, 175).

[5] J. Engel, J. Stueckler, and D. Cremers. "Large-Scale Direct SLAM with Stereo Cameras". In: *International Conference on Intelligent Robot Systems (IROS)*. 2015 (cited on pp. 11, 12, 14, 164).

[6] J. Engel, J. Sturm, and D. Cremers. "Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing". In: *Proceedings of the ICRA Workshop on Visual Control of Mobile Robots (ViCoMoR)*. 2012 (cited on pp. 9, 12).

[7] J. Engel, J. Sturm, and D. Cremers. "Camera-Based Navigation of a Low-Cost Quadrocopter". In: *International Conference on Intelligent Robot Systems (IROS)*. 2012 (cited on pp. 9, 12, 58, 112).

[8] J. Engel, J. Sturm, and D. Cremers. "Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera". In: *Robotics and Autonomous Systems (RAS)* 62.11 (2014), 1646—1656 (cited on pp. 9, 12, 140).

[9] J. Engel, J. Sturm, and D. Cremers. "Semi-Dense Visual Odometry for a Monocular Camera". In: *International Conference on Computer Vision (ICCV)*. 2013 (cited on pp. 11, 12, 14, 60, 64, 66, 67, 72, 76, 78–80, 83, 97, 98, 123, 125).

[10] J. Engel, V. Usenko, and D. Cremers. "A Photometrically Calibrated Benchmark For Monocular Visual Odometry". In: *arXiv:1607.02555*. 2016 (cited on pp. 11, 12, 16, 138, 150–153, 161).

[11] T. Schöps, J. Engel, and D. Cremers. "Semi-Dense Visual Odometry for AR on a Smartphone". In: *International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014 (cited on pp. 8, 11, 12, 14, 60, 145).

[12] V. Usenko, J. Engel, J. Stueckler, and D. Cremers. "Direct Visual-Inertial Odometry with Stereo Cameras". In: *International Conference on Robotics and Automation (ICRA)*. 2016 (cited on pp. 12, 14, 191).

[13] V. Usenko, J. Engel, J. Stueckler, and D. Cremers. "Reconstructing Street-Scenes in Real-Time From a Driving Car". In: *International Conference on 3D Vision (3DV)*. 2015 (cited on p. 12).

Bibliography

[14] M. Achtelik, S. Weiss, and R. Siegwart. "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments". In: *International Conference on Robotics and Automation (ICRA)*. 2011 (cited on pp. 58, 112).

[15] A. Akbarzadeh, J.M. Frahm, P. Mordohai, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér, and M. Pollefeys. "Towards Urban 3D Reconstruction from Video". In: *Proceedings of the International Symposium on 3D Data Processing, Visualization, and Transmission*. 2006 (cited on p. 59).

[16] S. Baker and I. Matthews. *Lucas-Kanade 20 Years On: A Unifying Framework*. Tech. rep. Carnegie Mellon University, 2002 (cited on pp. 53, 122).

[17] J.P. Barreto. "Unifying image plane liftings for central catadioptric and dioptric cameras". In: *Imaging Beyond the Pinhole Camera*. Springer, 2006, pp. 21–38 (cited on p. 117).

[18] H. Bay, T. Tuytelaars, and L.V. Gool. "SURF: Speeded up robust features". In: *European Conference on Computer Vision (ECCV)*. 2006 (cited on p. 4).

[19] S. Benhimane and E. Malis. "Real-Time Image-Based Tracking of planes using Efficient Second-order Minimization". In: *International Conference on Intelligent Robot Systems (IROS)*. 2004 (cited on p. 69).

[20] D.C. Brown. "The bundle adjustment – progress and prospects". In: *International Archives Photogrammetry* 21.3 (1976), pp. 1–1 (cited on p. 4).

[21] M. Brown and D. Lowe. "Automatic panoramic image stitching using invariant features". In: *International Journal of Computer Vision (IJCV)* 74.1 (2007), pp. 59–73 (cited on p. 167).

[22] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart. "The EuRoC micro aerial vehicle datasets". In: *International Journal of Robotics Research (IJRR)* (2016) (cited on pp. 150, 161, 167).

[23] A. Chiuso, P. Favaro, J. Hailin, and S. Soatto. "Structure from motion causally integrated over time". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 24.4 (2002), pp. 523–535 (cited on pp. 94, 112).

[24] J. Civera, A.J. Davison, and J. Montiel. "Inverse Depth Parametrization for Monocular SLAM". In: *Transactions on Robotics (TRO)* 24.5 (2008), pp. 932–945 (cited on p. 141).

[25] A. Clifford. *Multivariate Error Analysis*. John Wiley & Sons, 1973 (cited on p. 46).

[26] A. Comport, E. Malis, and P. Rives. "Accurate Quadri-focal Tracking for Robust 3D Visual Odometry". In: *International Conference on Robotics and Automation (ICRA)*. 2007 (cited on pp. 41, 59, 78, 94, 99, 100).

[27] A. Concha and J. Civera. "Using Superpixels in Monocular SLAM". In: *International Conference on Robotics and Automation (ICRA)*. 2014 (cited on p. 59).

[28] E. Coumans et al. *Bullet physics library,* `http://bulletphysics.org` (cited on p. 85).

[29] M. Cummins and P. Newman. "Appearance-only SLAM at large scale with FAB-MAP 2.0". In: *International Journal of Robotics Research (IJRR)* 30.9 (2011), pp. 1100–1123 (cited on p. 103).

[30] A.J. Davison and D.W. Murray. "Simultaneous Localization and Map-Building Using Active Vision". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 24.7 (2002), pp. 865–880 (cited on p. 94).

[31] A.J. Davison, I. Reid, N. Molton, and O. Stasse. "MonoSLAM: Real-time single camera SLAM". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 29.6 (2007), pp. 1052–1067 (cited on pp. 4, 6, 40, 77, 94, 112, 133, 134).

[32] P. Debevec and J. Malik. "Recovering high dynamic range radiance maps from photographs". In: *ACM SIGGRAPH classes*. 2008 (cited on pp. 167, 171).

[33] F. Devernay and O. Faugeras. "Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Enviroments". In: *Machine Vision and Applications* 13.1 (2001), pp. 14–24 (cited on p. 21).

[34] O. Dunkley. *Visual Inertial Control of a Nano-Quadrotor*. Master's Thesis. Technical University Munich. 2014 (cited on p. 9).

[35] E. Eade. *Lie Groups for 2D and 3D Transformations*. Tech. rep. Carnegie Mellon Univ., 2013 (cited on pp. 27, 28).

[36] E. Eade and T. Drummond. "Edge landmarks in monocular SLAM". In: *British Machine Vision Converence (BMVA)*. 2006 (cited on p. 59).

[37] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. "An Evaluation of the RGB-D SLAM System". In: *International Conference on Robotics and Automation (ICRA)*. 2012 (cited on p. 72).

[38] J. Engel. *Autonomous Camera-Based Navigation of a Quadrocopter*. Master's Thesis. Technical University Munich. 2011 (cited on p. 9).

[39] P. Erdős and A. Rényi. "On random graphs I." In: *Publicationes Mathematicae* 6 (1959), pp. 290 –297 (cited on p. 173).

[40] M. Fiala. "ARTag, a fiducial marker system using digital techniques". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005 (cited on p. 76).

[41] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *International Conference on Robotics and Automation (ICRA)*. 2014 (cited on pp. 10, 58, 60, 76, 135, 151).

[42] M.A. Föstner and E. Gülch. "A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features". In: *Proceedings of the ISPRS Intercommission Workshop*. 1987 (cited on p. 4).

[43] D. Gallup, J. Frahm, P. Mordohai, and M. Pollefeys. "Variable Baseline/Resolution Stereo". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008 (cited on pp. 41, 44).

[44] S. Garrido-Jurado, R. Munoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (2014), pp. 2280 –2292 (cited on p. 171).

[45] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (cited on p. 167).

[46] C. Geyer and K. Daniilidis. "A unifying theory for central panoramic systems and practical implications". In: *European Conference on Computer Vision (ECCV)*. 2000 (cited on pp. 117, 123).

[47] A. Glover, W. Maddern, M. Warren, R. Stephanie, M. Milford, and G. Wyeth. "OpenFABMAP: an open source toolbox for appearance-based loop closure detection". In: *International Conference on Robotics and Automation (ICRA)*. 2012 (cited on p. 69).

[48] D. Goldman and J. Chen. "Vignette and exposure calibration and compensation". In: *International Conference on Computer Vision (ICCV)*. 2005 (cited on p. 167).

[49] T. Goncalves and A.I. Comport. "Real-time Direct Tracking of Color Images in the Presence of Illumination Variation". In: *International Conference on Robotics and Automation (ICRA)*. 2011 (cited on p. 95).

[50] D. Gutierrez, A. Rituerto, J.M. Montiel, and J.J. Guerrero. "Adapting a real-time monocular visual SLAM from conventional to omnidirectional cameras". In: *Proceedings of the ICCV Computer Vision Workshops*. 2011 (cited on p. 114).

[51] A. Handa, R. Newcombe, A. Angeli, and A. Davison. "Real-Time Camera Tracking: When Is High Frame-Rate Best?" In: *European Conference on Computer Vision (ECCV)*. 2012 (cited on pp. 70, 72).

[52] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM". In: *International Conference on Robotics and Automation (ICRA)*. 2014 (cited on pp. 150, 161, 167).

[53] K.J. Hanna. "Direct multi-resolution estimation of ego-motion and structure from motion". In: *Proceedings of the IEEE Workshop on Visual Motion*. 1991 (cited on p. 4).

[54] C. Harris and M. Stephens. "A combined corner and edge detector". In: *Proceedings of the Fourth Alvey Vision Conference*. 1988, pp. 147–151 (cited on pp. 4, 41).

[55] B. Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *Journal of the Optical Society of America* 4.4 (1987), pp. 629–642 (cited on p. 69).

[56] G.P. Huang, A.I. Mourikis, and S.I. Roumeliotis. "A first-estimates Jacobian EKF for improving SLAM consistency". In: *International Symposium on Experimental Robotics (ISER)*. 2008 (cited on p. 142).

[57] M. Irani and P. Anandan. "All About Direct Methods". In: *Proceedings of the ICCV International Workshop on Vision Algorithms: Theory and Practice*. 1999 (cited on pp. 107, 108).

[58] H. Jin, P. Favaro, and S. Soatto. "A semi-direct approach to structure from motion". In: *The Visual Computer* 19.6 (2003), pp. 377–394 (cited on p. 5).

[59] H. Jin, P. Favaro, and S. Soatto. "Real-time 3-d motion and structure of point features: Front-end system for vision-based control and interaction". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000 (cited on p. 4).

[60] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *International Journal of Robotics Research (IJRR)* 31.2 (2012), pp. 217–236 (cited on p. 162).

[61] O. Kahler and J. Denzler. "Tracking and reconstruction in a combined optimization approach". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34.2 (2012), pp. 387–401 (cited on pp. 78, 80).

[62] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. *Digital video stabilization and rolling shutter correction using gyroscopes*. Tech. rep. Stanford University, 2011 (cited on p. 83).

[63] C. Kerl, M. Souiai, J. Sturm, and D. Cremers. "Towards Illumination-invariant 3D Reconstruction using ToF RGB-D Cameras". In: *International Conference on 3D Vision (3DV)*. 2014 (cited on p. 167).

[64] C. Kerl, J. Stueckler, and D. Cremers. "Dense Continuous-Time Tracking and Mapping with Rolling Shutter RGB-D Cameras". In: *International Conference on Computer Vision (ICCV)*. 2015 (cited on p. 159).

[65] C. Kerl, J. Sturm, and D. Cremers. "Dense Visual SLAM for RGB-D Cameras". In: *International Conference on Intelligent Robot Systems (IROS)*. 2013 (cited on pp. 59, 60, 63, 72, 78, 94, 164).

[66] C. Kerl, J. Sturm, and D. Cremers. "Robust Odometry Estimation for RGB-D Cameras". In: *International Conference on Robotics and Automation (ICRA)*. 2013 (cited on pp. 40, 41, 43, 52, 54, 94).

[67] S. Kim and M. Pollefeys. "Robust radiometric calibration and vignetting correction". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30.4 (2008), pp. 562–576 (cited on p. 167).

[68] G. Klein and D. Murray. "Improving the agility of keyframe-based SLAM". In: *European Conference on Computer Vision (ECCV)*. 2008 (cited on pp. 41, 59).

[69] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *International Symposium on Mixed and Augmented Reality (ISMAR)*. 2007 (cited on pp. 40, 42, 54, 58, 72, 77, 87, 94, 112, 134, 169).

[70] G. Klein and D. Murray. "Parallel Tracking and Mapping on a Camera Phone". In: *International Symposium on Mixed and Augmented Reality (ISMAR)*. 2009 (cited on pp. 76, 77).

[71] S. Klose, P. Heise, and A. Knoll. "Efficient Compositional Approaches for Real-Time Robust Direct Visual Odometry from RGB-D Data". In: *International Conference on Intelligent Robot Systems (IROS)*. 2013 (cited on pp. 69, 95).

[72] E. Kruppa. "Zur Ermittlung eines Objekts aus zwei Perspektiven mit innerer Orientierung". In: *Sitzungsberichte der math.-naturw. Kl. der kaiserlichen Akademie der Wissenschaften, Abt. IIa* 122 (1913), pp. 1939–1948 (cited on p. 4).

[73] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A General Framework for Graph Optimization". In: *International Conference on Robotics and Automation (ICRA)*. 2011 (cited on pp. 60, 63, 69).

[74] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization". In: *Robotics: Science and Systems (RSS)*. 2013 (cited on p. 112).

[75] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *International Journal of Robotics Research (IJRR)* 34.3 (2015), pp. 314–334 (cited on pp. 11, 136, 141–143, 155).

[76] M. Li, B. Kim, and A. Mourikis. "Real-Time Motion Estimation on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera". In: *International Conference on Robotics and Automation (ICRA)*. 2013 (cited on pp. 83, 159).

[77] M. Li, B. Hyung Kim, and A.I. Mourikis. "Real-time Motion Tracking on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera". In: *International Conference on Robotics and Automation (ICRA)*. 2013 (cited on p. 11).

[78] M. Li and A. Mourikis. "High-Precision, Consistent EKF-based Visual-Inertial Odometry". In: *International Journal of Robotics Research (IJRR)* 32.6 (2013), pp. 690–711 (cited on pp. 58, 77, 112).

[79] S. Lovegrove. *Parametric Dense Visual SLAM*. PhD Thesis. Imperial College London. 2012 (cited on p. 28).

[80] D.G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *International Conference on Computer Vision (ICCV)*. 1999 (cited on p. 4).

[81] L. Matthies, R. Szeliski, and T. Kanade. "Incremental estimation of dense depth maps from image Image Sequences". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 1988 (cited on pp. 4, 42).

[82] J. Maye, P. Furgale, and R. Siegwart. "Self-supervised Calibration for Robotic Systems". In: *Proceedings of the Intelligent Vehicles Symposium*. 2013 (cited on p. 125).

[83] M. Meilland and A.I. Comport. "On unifying key-frame and voxel-based dense visual SLAM at large scales". In: *International Conference on Intelligent Robot Systems (IROS)*. 2013 (cited on p. 94).

[84] M. Meilland, A.I. Comport, and P. Rives. "A spherical robot-centered representation for urban navigation". In: *International Conference on Intelligent Robot Systems (IROS)*. 2010, pp. 5196–5201 (cited on p. 114).

[85] A.I. Mourikis and S.I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *International Conference on Robotics and Automation (ICRA)*. 2007 (cited on pp. 10, 76).

[86] R. Mur-Artal, J. Montiel, and J. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *Transactions on Robotics (TRO)* 31.5 (2015), pp. 1147–1163 (cited on pp. 10, 134, 151, 164, 168, 179).

[87] R. Newcombe. *Dense Visual SLAM*. PhD Thesis. Imperial College London. 2012 (cited on p. 10).

[88] R. Newcombe, S. Lovegrove, and A. Davison. "DTAM: Dense tracking and mapping in real-time". In: *International Conference on Computer Vision (ICCV)*. 2011 (cited on pp. 9, 40–43, 59, 76, 78, 112, 133–135, 164).

[89] D. Nister, O. Naroditsky, and J. Bergen. "Visual odometry". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2004 (cited on p. 94).

[90] M. Okutomi and T. Kanade. "A Multiple-Baseline Stereo". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 15.4 (1993), pp. 353–363 (cited on pp. 42, 44).

[91] I. Palasti. "On the connectedness of bichromatic random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci.* 8 (1963), pp. 341 –440 (cited on p. 173).

[92] L.M. Paz, P. Pinies, J.D. Tardos, and J. Neira. "Large-Scale 6-DOF SLAM With Stereo-in-Hand". In: *Transactions on Robotics (TRO)* 34.5 (2008), pp. 946–957 (cited on p. 94).

[93] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *International Conference on Robotics and Automation (ICRA)*. 2014 (cited on pp. 59, 76, 78, 133, 135, 164).

[94] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. "Detailed Real-Time Urban 3D Reconstruction from Video". In: *International Journal of Computer Vision (IJCV)* 78.2-3 (2008), pp. 143–167 (cited on pp. 41, 44).

[95] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. "Dense Monocular Depth Estimation in Complex Dynamic Scenes". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on p. 134).

[96] A. Rituerto, L. Puig, and J.J. Guerrero. "Comparison of omnidirectional and conventional monocular systems for visual SLAM". In: *Proceedings of the Workshop on Omnidirectional Vision, Camera Networks and Nonclassical Cameras (OMNIVIS)*. 2010 (cited on p. 114).

[97] E. Rosten, R. Porter, and T. Drummond. "FASTER and better: A machine learning approach to corner detection". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 32.1 (2010), pp. 105–119 (cited on p. 4).

[98] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: an efficient alternative to SIFT or SURF". In: *International Conference on Computer Vision (ICCV)*. 2011 (cited on p. 4).

[99] J. Prankl S. Alexandrov and M. Vincze. "Color Correction for 3D Mapping with RGB-D Cameras". In: *International Conference on Intelligent Robot Systems (IROS)*. 2016 (cited on pp. 167, 171).

[100] T. Sato, M. Kanbara, N. Yokoya, and H. Takemura. "Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held camera". In: *International Journal of Computer Vision (IJCV)* 47.1-3 (2002), pp. 1–3 (cited on p. 41).

[101] O. Saurer, K. Köser, J.Y. Bouguet, and M. Pollefeys. "Rolling Shutter Stereo". In: *International Conference on Computer Vision (ICCV)*. 2013 (cited on p. 83).

[102] D. Scaramuzza, A. Martinelli, and R. Siegwart. "A flexible technique for accurate omnidirectional camera calibration and structure from motion". In: *International Conference on Computer Vision Systems*. 2006 (cited on p. 117).

[103] D. Scaramuzza and R. Siegwart. "Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles". In: *Transactions on Robotics (TRO)* 24.5 (2008), pp. 1015–1026 (cited on p. 114).

[104] T. Schöps. *Semi-dense visual SLAM on mobile devices*. Master's Thesis. Technical University Munich. 2014 (cited on p. 8).

[105] C. Silpa-Anan and R. Hartley. "Visual localization and loop-back detection with a high resolution omnidirectional camera". In: *Workshop on Omnidirectional Vision*. 2005 (cited on p. 114).

[106] G. Silveira, E. Malis, and P. Rives. "An efficient direct approach to visual SLAM". In: *Transactions on Robotics (TRO)* 24.5 (2008), pp. 969–979 (cited on p. 78).

[107] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD Thesis. Imperial College London. 2012 (cited on p. 28).

[108] H. Strasdat, A.J. Davison, J.M.M. Montiel, and K. Konolige. "Double window optimisation for constant time visual SLAM". In: *International Conference on Computer Vision (ICCV)*. 2011 (cited on pp. 112, 162).

[109] H. Strasdat, J. Montiel, and A. Davison. "Scale Drift-Aware Large Scale Monocular SLAM." In: *Robotics: Science and Systems (RSS)*. 2010 (cited on pp. 60, 62, 63, 67, 94).

[110] H. Strasdat, J.M.M. Montiel, and A.J. Davison. "Real-time monocular SLAM: Why filter?" In: *International Conference on Robotics and Automation (ICRA)*. 2010 (cited on p. 7).

[111] J. Stückler and S. Behnke. "Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video". In: *International Journal of Computer Vision (IJCV)* 113.3 (2015), pp. 233–245 (cited on p. 108).

[112] J. Stuehmer, S. Gumhold, and D. Cremers. "Real-Time Dense Geometry from a Handheld Camera". In: *Pattern Recognition (DAGM)*. 2010 (cited on pp. 10, 40, 41, 43, 59, 78, 89, 133–135).

[113] L. von Stumberg. *Autonomous Flight of a Low-Cost Quadrocopter using a Semi-Dense Monocular SLAM System*. Bachelor's Thesis. Technical University Munich. 2015 (cited on p. 9).

[114] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *International Conference on Intelligent Robot Systems (IROS)*. 2012 (cited on pp. 42, 54, 70–72, 87, 167).

[115] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco Paulsen, O. Saurer, and M. Pollefeys. "Live Metric 3D Reconstruction on Mobile Phones". In: *International Conference on Computer Vision (ICCV)*. 2013 (cited on p. 77).

[116] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. "Monocular visual odometry in urban environments using an omnidirectional camera". In: *International Conference on Intelligent Robot Systems (IROS)*. 2008 (cited on p. 114).

[117] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. Tech. rep. Carnegie Mellon University, 1991 (cited on p. 4).

[118] T.M. Tykkala and A.I. Comport. "A dense structure model for image based stereo SLAM". In: *International Conference on Robotics and Automation (ICRA)*. 2011 (cited on p. 94).

[119] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. "Dense versus sparse approaches for estimating the fundamental matrix". In: *International Journal of Computer Vision (IJCV)* 96.2 (2012), pp. 212–234 (cited on p. 134).

[120] D. Wagner, T. Langlotz, and D. Schmalstieg. "Robust and unobtrusive marker tracking on mobile phones". In: *International Symposium on Mixed and Augmented Reality (ISMAR)*. 2008 (cited on p. 76).

[121] D. Wagner and D. Schmalstieg. "ARToolKitPlus for pose tracking on mobile devices". In: *Proceedings of 12th Computer Vision Winter Workshop*. 2007 (cited on p. 76).

[122] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. "Dense reconstruction on-the-fly". In: *European Conference on Computer Vision (ECCV)*. 2012 (cited on pp. 40, 41).

[123] X. Ying and Z. Hu. "Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model". In: *European Conference on Computer Vision (ECCV)*. 2004 (cited on p. 117).

[124] G. Zhang, J. Jia, and H. Bao. "Simultaneous Multi-Body Stereo and Segmentation". In: *International Conference on Computer Vision (ICCV)*. 2011 (cited on p. 108).