Local Tracking and Mapping for Direct Visual SLAM

Scientific work for obtaining the academic degree
Master of Science (M.Sc.) in Mechanical Engineering
at the Department of Mechanical Engineering of the Technical University of Munich

Supervisor	Prof. Dr.-Ing. Manfred Hajek Chair of Helicopter Technology
Advisors	Nikolaus Demmel, M.Sc. Computer Vision Group – Department of Informatics
	Tim Mehling, M.Sc. Chair of Helicopter Technology
Submitted by	Pablo Rodríguez Palafox
Submission date	October 14, 2019 in Garching bei München

Statement of Authorship

I hereby declare that the work presented in this thesis has been performed and interpreted solely by myself, using no other than the specified sources and resources.

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Garching, October 14, 2019 Pablo Rodríguez Palafox

Acknowledgements

I would like to start thanking Prof. Dr. Daniel Cremers and Prof. Dr.-Ing. Manfred Hajek for offering me the possibility to pursue my Master's Thesis as a collaboration project between the Computer Vision Group of TUM's Department of Informatics and the Chair of Helicopter Technology of TUM's Department of Mechanical Engineering. Additionally, I would like to express my sincere gratitude to Nikolaus Demmel and Tim Mehling for their continuous support and many fruitful discussions along the project. Finally, I would like to thank Vladyslav Usenko and Hide Matsuki for their valuable insights and recommendations.

Abstract

Recently, direct approaches to Visual Odometry (VO) have shown great accuracy and robustness, outperforming feature-based methods in some scenarios [13]. Such direct VO systems have been combined with both feature-based place recognition techniques and pose-graph optimization in order to reduce the accumulated drift when revisiting already mapped areas [20]. However, in this line of work the front-end odometry estimator does not incorporate the updated map information. Inspired by the recent Direct Sparse Mapping (DSM) [64], in this thesis we present a direct SLAM system that localizes the camera in a local map similarly to ORB-SLAM's [47] local tracking and mapping, but leveraging direct image alignment and photometric bundle adjustment instead of features. We implement and describe in detail all major components, including direct image alignment (in its forward additive and inverse compositional variants), map point selection and depth initialization, keyframe and landmark management (in particular the reuse of previously mapped regions to ensure a consistent representation of the scene), and local window Photometric Bundle Adjustment (PBA), which are combined into a modular and flexible direct SLAM framework. A coarse-to-fine scheme and a robust error function (based on the t-distribution) further ensure the stability of both tracking and PBA. Additionally, our system is not limited to the pinhole model. Unlike other existing implementations, it can work directly on distorted images, thus being suitable for fish-eye lenses, much like Omnidirectional DSO [42]. Experiments on realistic synthetic data from a game-engine-based simulator with both pose and structure ground-truth allow us to validate the individual components of the system as well as the overall performance. With this thesis we contribute a proof of concept for a direct SLAM system that locally reuses map point information, as well as an extensible framework for further research in this field.

Contents

Abbreviations and Symbols

Acronyms

3D	3-dimensional
ARI	Average Runtime per Iteration
ATE	Absolute Trajectory Error
BA	Bundle Adjustment
BoW	Bag-of-Words
CPU	Central Processing Unit
DIA	Direct Image Alignment
DoF	Degree of Freedom
DSCM	Double Sphere Camera Model
EKF	Extended Kalman Filter
EUCM	Extended Unified Camera Model
FA	Forward Additive
G-N	Gauss-Newton
GPU	Graphical Processing Unit
i.i.d.	independent and identically distributed
IC	Inverse Compositional
IMU	Inertial Measurement Unit
IRLS	Iteratively Reweighted Least Squares
L-M	Levenberg-Marquardt
LC	Loop Closure
LCW	Local Covisibility Window
MAD	Median Absolute Deviation
PBA	Photometric Bundle Adjustment

PDF	Probability Density Function
PGO	Pose-Graph Optimization
RANSAC	Random Sample Consensus
RGB-D	Red-Green-Blue-Depth
RPE	Relative Pose Error
SLAM	Simultaneous Localization and Mapping
SSD	Sum over Squared Distances
TR	Total Runtime
UAV	Unmanned Aerial Vehicle
UCM	Unified Camera Model
VIO	Visual Inertial Odometry
VO	Visual Odometry

List of Figures

List of Tables

1. Introduction

Simultaneous Localization and Mapping (SLAM) and Visual Odometry (VO) are key components of many emerging technologies, ranging from autonomous cars and Unmanned Aerial Vehicles (UAVs) to virtual and augmented reality. In essence, both try to recover the camera poses and a map of the viewed scene from a video feed in real time.

VO systems only build a temporary map to track the subsequent camera poses. They typically use a sliding-window of active *keyframes* –selected frames that provide more and better information than merely using consecutive frames– and marginalize map points and keyframes that leave the field of view. Consequently, if the camera revisits previously mapped areas, already marginalized points and keyframes cannot be reused and the system needs to duplicate them. This can cause motion drift and map inconsistencies.

In contrast, SLAM methods aim to build a persistent map of the scene by processing map point reobservations. Therefore, map points (which we will also refer to as *landmarks*) and keyframes that leave the field of view are not marginalized but rather retained and marked as inactive. Such map points and keyframes can be reactivated later on according to covisibility criteria. (Note that two keyframes are in general considered covisible if they observe several landmarks in common.) Explicitly managing map point reobservations and building a network of keyframes connected not only temporally but also in terms of the scene region they view can greatly help reduce motion drift. Figure 1.1 exemplifies this key insight of reusing landmarks instead of duplicating them when revisiting previously mapped areas.

Traditionally, feature-based or indirect methods have dominated the field of SLAM and VO. However, recent direct approaches [13] have shown great accuracy and robustness, even outperforming feature-based systems in some scenarios. Typically, both indirect and direct methods take a probabilistic approach, where noisy observations $\hat{\boldsymbol{f}}$ are taken as input and used to compute an estimator $\boldsymbol{\theta}$ for the unknown model parameters (camera poses and world structure). Employing a Maximum Likelihood approach one can find the model parameters $\boldsymbol{\theta}$ that maximize the probability of obtaining the measurements $\hat{\boldsymbol{f}}$, i.e., $\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}} p(\hat{\boldsymbol{f}}|\boldsymbol{\theta})$.

Indirect methods preprocess the raw sensor measurements to produce an intermediate representation, commonly denoted as *features*. This is typically approached by extracting a set of sparse keypoints (e.g., Harris corners [25]) from the image and matching them across different frames to establish correspondences. Direct approaches, on the other hand, use the actual sensor values – light intensity striking one of the several tiny camera sensors from a certain direction over a certain time period – as observations $\hat{\boldsymbol{f}}$ in this probabilistic model. Therefore, instead of being limited to corners, direct approaches can sample information from across all available image

(a) VO **(b)** SLAM

Figure 1.1.: Duplication of map points (which we will also refer to as landmarks) in VO versus their reuse in SLAM systems. In (a), already mapped points hosted in, i.e., relative to, an old keyframe (orange camera) are duplicated when the sensor at the current position (blue camera) revisits the region. Thus, some landmarks (blue) will be mere duplicates of previously mapped landmarks (orange), which is how VO systems typically function. In contrast, SLAM methods attempt to reuse landmark information to better localize the sensor within the map (b), therefore avoiding point duplication.

data, including edges and surfaces with low intensity variation. This increases robustness in environments with little to no texture, such as walls.

Moreover, current direct approaches have been combined with both feature-based place recognition techniques –well-known from indirect SLAM methods such as ORB-SLAM [47]– and subsequent Pose-Graph Optimization (PGO) to reduce the accumulated drift when revisiting a previously mapped area [20]. However, in this line of work the front-end odometry estimator does not incorporate the updated map information.

1.1. Contribution

In this thesis we investigate a direct SLAM approach that localizes the camera in a local map similar to ORB-SLAM's local tracking and mapping, but leveraging Direct Image Alignment (DIA) and Photometric Bundle Adjustment (PBA) instead of features. To this end, and inspired by the recent DSM by Zubizarreta et al. [64], we retain map points and keyframes that leave the field of view and re-activate them according to covisibility criteria, effectively building a direct SLAM system.

We implement and describe in detail all major components of this direct SLAM, including DIA [3, 30] (in its forward additive and inverse compositional variants), point selection and initialization, keyframe and landmark (map point) management (in particular the reuse of previously mapped regions to ensure a consistent representation of the scene), and local window PBA, which are combined into a modular and flexible direct SLAM framework.

Moreover, we implement a coarse-to-fine scheme and a robust error function (based on the fitting

of the t-distribution to the photometric residuals) to ensure the stability of both camera tracking (DIA) and PBA. Furthermore, our implementation for such time-consuming modules, i.e., camera tracking and PBA, is highly parallelized.

Additionally, the system we develop in this thesis is not limited to the pinhole model. Unlike other existing implementations, such as DSO [13] or DSM [64], our direct SLAM can work directly on distorted images, thus being suitable for fisheye lenses, much like Omnidirectional DSO [42].

Finally, our implementation is highly modular and flexible, allowing the user, for instance, to select between four different camera models, two solvers for camera tracking or two PBA implementations (either using a Ceres-based [1] solver or a custom solver implemented by ourselves.)

Experiments on realistic synthetic data from a game-engine-based simulator with both pose and structure groundtruth allow us to validate the individual components of the system as well as the overall performance of our method. With this thesis we contribute a proof of concept for a direct SLAM system that locally reuses map point information, as well as an extensible framework for further research in this field.

1.2. Outline

This work is organized as follows. In Chapter 2 we present an overview of the state-of-the-art in both direct and indirect SLAM and VO systems. Chapter 3 introduces basic theoretical concepts that we use throughout the thesis. We give an overview of the developed approach in Chapter 4 and further delve into the proposed system in Chapters 5 and 6. Experimental results are presented in Chapter 7. Finally, Chapter 8 concludes the thesis and provides an outlook on possible future lines of work.

2. Related Work

Typically, a real-time SLAM or VO system consists of two separate threads, namely tracking and mapping. The first approaches to SLAM were indirect or feature-based. Recently, direct monocular SLAM methods have been proposed.

2.1. Indirect Monocular SLAM and VO

Feature-based methods split the overall problem –recovering scene structure and camera poses from images– into two steps: first, a set of salient image features is extracted from the image and matched to other features in previous frames using invariant feature descriptors; second, based on this correspondence matching the camera motion and the scene geometry can be computed through epipolar geometry and the minimization of the reprojection error.

This decoupling and in particular the preliminary abstraction step render the resulting system both less complex and real-time capable. However, this also reduces information to the feature type employed, typically corners and blobs [25]. Information contained in straight or curved edges, ubiquitous in man-made environments, is thus discarded. Even though edge-based [34, 12] and region-based [9] features have been proposed to overcome this limitation, the complexity in the estimation of the high-dimensional feature space renders such approaches difficult to use in practice. Moreover, feature-based SLAM and VO systems rely on the quality of feature detectors and matching strategies, mostly optimized for speed rather than precision. This causes drift in the motion estimate, which needs to be compensated through the use of costly outlier estimation methods such as Random Sample Consensus (RANSAC) [16].

Several works have been proposed that leverage features for the problem of SLAM. Davison in [10] presents a real-time monocular SLAM system (MonoSLAM) within an Extended Kalman Filter (EKF)-based framework. Civera et al. in [6] extend MonoSLAM and include an inverse parametrization of depth, which we also employ in our work. A cornerstone in visual SLAM was proposed by Klein and Murray in PTAM [35], where the tasks of tracking and mapping are parallelized for the first time into two separate threads, thus demonstrating the feasibility of maintaining a persistent map of the scene through the use of Bundle Adjustment (BA). In [47] Mur-Artal et al. present ORB-SLAM, a full SLAM system that puts together traditional geometric BA with map reuse capability, Loop Closure (LC) and a relocalization module. Its management of map point reobservations in the BA is arguably its most important feature, which makes it up to date one of the most accurate monocular SLAM methods in several scenarios.

Regarding VO, OKVIS [38] proposes a feature-based Visual Inertial Odometry (VIO) system that achieves real-time operation by limiting optimization to a bounded window of keyframes, marginalizing geometry and cameras that fall outside this local window.

2.2. Direct Monocular SLAM and VO

Direct methods [27] work directly with the intensity values of the image pixels instead of extracting intermediate features. This allows them to exploit all the information contained in the image –including edges and weak intensity variations–, and have been shown to outperform feature-based approaches in therms of robustness in sparsely textured environments [39]. Furthermore, the more costly computation of the photometric error in comparison to the reprojection error is compensated by not having to compute features or invariant descriptors.

The first direct monocular SLAM methods relied on tracking and mapping planar patches [28, 46, 53, 43] to estimate structure and motion, either by taking a filtering approach [28, 46] or by employing nonlinear least squares optimization [53, 43, 50], but in all cases estimating the surface normals of the patches. Real-time performance in these works is only possible by tracking a few selected planar regions and on small datasets, as reported by the authors in [46, 53, 43]. In [7] and later extended in [8], Comport et al. proposed a stereo-based VO system where the local planarity assumption is relaxed and direct tracking is done with respect to arbitrary 3-dimensional (3D) structures.

For Red-Green-Blue-Depth (RGB-D) sensors, DIA is well established, both as VO [32, 60, 45] or full SLAM systems [31]. However, only more recently monocular direct VO algorithms have appeared that perform in real time. In [56, 48, 49], fully dense depth maps are computed for every keyframe through a variational formulation, i.e., by minimizing a global and spatially-regularized energy functional. New frames are tracked through direct whole-image alignment using the depth map of previous keyframes. These approaches, however, require a state-of-the-art Graphical Processing Unit (GPU) to run in real time.

To eliminate the need of a GPU, a semi-dense depth filtering formulation was proposed in [15], which allows for real-time operation on a Central Processing Unit (CPU) or even on a smartphone [52] by only using pixels characterized with a strong gradient. In SVO [17] a hybrid semi-direct monocular VO is presented where new points are tracked using the photo-consistency assumption and then optimized using the reprojection error, achieving high frame rates on embedded platforms.

More recently, Engel et al. [13] presented DSO, the first fully direct VO method based on the joint optimization of all involved parameters (camera intrinsics and extrinsics plus geometry, represented as inverse depth in a reference frame) over a window of recent keyframes. Within this PBA formulation, a photometrically calibrated model for image formation is also taken into account. In DSO, keyframes and points that leave the field of view of the latest camera position are marginalized, following the approach presented by OKVIS [38].

All of the monocular direct approaches described above are pure visual odometries, where the motion of the camera is only tracked locally and map points are not reused by means of loop-closures. This causes inconsistencies in the map and, eventually, drift in the camera motion estimation. To address this issue, VO systems are commonly extended in the following manner. First, a feature-based place recognition algorithm detects LCs between keyframes. Then, a pose-graph is constructed where keyframes are represented as vertices and the transforms between keyframes as edges or constraints. The optimization of such a pose-graph using a generic graph optimization framework like g2o [36] produces a consistent, global map.

In LSD-SLAM [14], large-scale loop-closure keyframes are detected using FAB-MAP [24], an appearance-based mapping algorithm. Moreover, the inherent scale-ambiguity of monocular SLAM is accounted for by representing camera poses as 3D similarity transforms instead of rigid body motion, similarly to what [55] proposes. LSD-SLAM was the first scale-aware direct monocular SLAM method for large-scale environments. In a similar fashion, LDSO [20] extends DSO with a conventional ORB-based Bag-of-Words (BoW) [19] module, also employing g2o [36] for graph optimization.

While LSD-SLAM and LDSO are capable of dealing with LCs and thus reducing motion and scale drift considerably, they still present the following limitations: (1) LC detection relies on feature repeatability, thus missing many corrections; (2) the objective functions of the odometry and the pose-graph optimization differ; (3) even though the trajectory is spatially corrected, map points are not reused, but rather duplicated when revisiting already mapped areas. The recently proposed DSM [64] addresses these issues by reusing map points to build a persistent representation of the scene. Similarly to ORB-SLAM, reobservations in DSM are processed within the local-window PBA, which allows for more accurate estimates. However, DSM only works for the pinhole model and does not include a manual solver for PBA. Moreover, its code is not yet publicly available, unlike that of DSO [13] or LDSO [20].

Table 2.1 summarizes the contributions of our approach and compares its features with the properties of state-of-the-art systems. In a nutshell: (1) our direct SLAM method is built in a modular fashion, so as to enable future rapid extensions, much like a framework. This compares positively with other systems, whose code is less accessible in terms of readability, thus being more complex to extend and modify. The lack of such a modular, flexible and extensible framework has largely motivated this thesis; (2) including a custom solver for the time-consuming PBA problem and (3) allowing to easily select between different camera models differentiates our work from DSM [64]; (4) moreover, as opposed to [13, 20, 42], we include a coarse-to-fine approach, (5) the use of the t-distribution as a robust weight estimator for the PBA, and (6) the reuse of map point reobservations to ensure a consistent map, as also recently proposed in DSM [64].

	Code available	Modular Code	Fisheye lenses	Custom PBA solver	Coarse-to-fine PBA	Robust influence function	Marginalization	Local map reuse	Global LC / PGO
DSO	X			X			X		
Omni DSO			X	X			X		
LDSO	X			X			X		X
DSM		?			X	X		X	
Ours	X	X	X	X	X	X		X	

Table 2.1.: Comparison of the contributions and properties of several direct systems. The properties compared are as follows. **Code available:** whether the code is publicly available; **Modular code:** whether the code can be easily extended with new features or modules; **Fisheye lenses:** whether the system can directly be deployed on sequences recorded with fisheye lenses; **Custom PBA solver:** whether the system implements a custom solver for the costly PBA problem; **Custom PBA solver:** whether a coarse-to-fine optimization scheme is employed in the PBA in order to account for large displacements by increasing the convergence radius of PBA; **Robust influence function**: whether a robust influence function based on the actual residual distribution (in contrast to simply using the Huber norm) is employed to down-weight the influence of outliers in the PBA; **Marginalization**: whether marginalization is employed to remove variables in order to maintain the set of variables below a certain size; **Local map reuse**: whether old map points are reused (in contrast to duplicating them) when revisiting a previously mapped region; and **Global** LC / PGO: whether global loop closures are detected and corrected by means of a pose-graph optimization.

3. Preliminaries

In this chapter, we first present the notation used throughout the thesis, as well as the parametrization we use for the landmarks (or map points) in our map. The models for the image formation process (Section 3.3) and for motion representation (Section 3.4) are also described. A brief introduction to the least squares technique for parameter estimation is given in Section 3.5. Finally, additional miscellaneous background is given in Section 3.6.

3.1. Notation

Throughout this work, light lowercase letters (u) denote scalars or functions, matrices are represented by bold uppercase letters ($\mathbf{R}$) and bold lowercase letters represent vectors ($\mathbf{t}$).

3.2. Landmark Parametrization

We employ the inverse distance parametrization [6] for our landmarks (or map points). Instead of parametrizing a landmark by its 3D coordinates $\mathbf{p}$, we defined it by its (unit-length) bearing vector

$$\mathbf{b} = \frac{\mathbf{p}}{\|\mathbf{p}\|}, \tag{3.1}$$

which, as $\mathbf{p}$, is defined relative to the landmark's host frame (i.e., the camera frame in which the landmark "lives"), and by the inverse distance

$$d_{\mathbf{p}} = \frac{1}{\|\mathbf{p}\|} \tag{3.2}$$

from the point to the camera's center of projection. This formulation allows us to write the 3D coordinates of the landmark as follows:

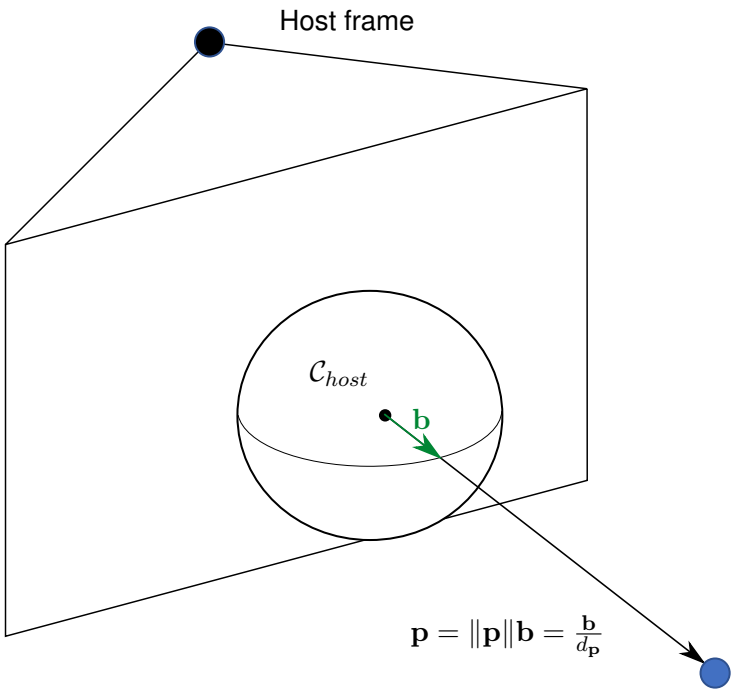


Figure 3.1.: Inverse Distance Parametrization. We parametrize a 3D point by its bearing vector $\mathbf{b} = \frac{\mathbf{p}}{\|\mathbf{p}\|}$ relative to the point's host frame (where the point "resides") and its inverse distance $d_\mathbf{p} = \frac{1}{\|\mathbf{p}\|}$.

$$\mathbf{p} = \frac{\mathbf{b}}{d_\mathbf{p}}. \tag{3.3}$$

Figure 3.1 visualizes the relation between the different variables.

In Equation (3.3), if $d_\mathbf{p}$ is zero or close to zero then we refer to $\mathbf{p}$ as a point at infinity. In a practical sense, a point at infinity can be a point on the horizon. (As will become apparent below, such a point cannot be used to estimate translation, but does help in estimating rotation.) We describe how to deal with such points towards the end of Section 3.4.1.

3.3. Calibration

The complete image formation process comprises a *geometric* and a *photometric* camera model. While the former projects a 3D point onto the 2D image plane, the latter maps real-world energy gathered by a pixel on the sensor, i.e., irradiance, to an intensity value. In indirect approaches, where feature extractors and descriptors are designed to be highly robust to photometric noise, the geometric model suffices, whereas in direct approaches a photometric model can be greatly beneficial, as reported in [13]. For simplicity, in this work we only employ a geometric camera model. However, a future extension that includes the photometric model analogously to DSO [13] or DSM [64] is possible in a straight forward manner.

3.3.1. Geometric Camera Calibration

Given the intrinsic parameters $\mathbf{c}$ of a camera, the projection function describes the mapping

$$\pi_{\mathbf{c}} : \mathbb{R}^3 \to \Omega \subset \mathbb{R}^2; \quad \mathbf{p} \mapsto \mathbf{u} \tag{3.4}$$

of a 3D point $\mathbf{p} = (x, y, z)^\top \in \mathbb{R}^3$ into the 2D image plane $\Omega \subset \mathbb{R}^2$, where pixel coordinates are denoted as $\mathbf{u} = (u, v)^\top \in \Omega$ [26]. In turn, we can define the camera's inverse projection function

$$\pi_{\mathbf{c}}^{-1} : \Omega \to \mathbb{R}^3; \quad \mathbf{u} \mapsto \mathbf{b}(\mathbf{u}) \tag{3.5}$$

as an unprojection of the image coordinates $\mathbf{u}$ to the bearing vector of unit-length

$$\mathbf{b} = \pi_{\mathbf{c}}^{-1}(\mathbf{u}) \in \mathbb{R}^3, \tag{3.6}$$

which corresponds to the same vector as that presented in Equation (3.1). It is important to notice that $\mathbf{b}$ implicitly defines the ray of all the 3D points that project into the same image location.

Various geometric camera models exist. In the remainder of this section we present the projection and unprojection functions for the well-known pinhole model, the Extended Unified Camera Model (EUCM) and the Double Sphere Camera Model (DSCM). We refer the reader to [61] for a more in-depth study of different geometric camera models.

Pinhole Camera Model

The pinhole camera model abstracts the camera to an infinitely small hole, the pinhole, and the image plane. Only light rays that fall through the pinhole and intersect the image plane get projected. This model has four intrinsic parameters, $\mathbf{c} = (f_x, f_y, c_x, c_y)^\top$, and we can define the projection function as follows:

$$\pi_{\mathbf{c}}(\mathbf{p}) = \begin{pmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}. \tag{3.7}$$

To unproject a 2D point back to 3D we can use the following function:

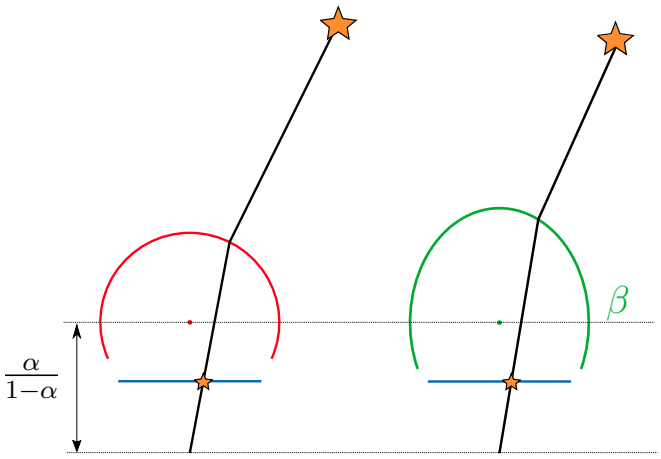


Figure 3.2.: Schematic representation of the Unified and the Extended Unified Camera Models. [61]

$$\pi_{\mathbf{c}}^{-1}(\mathbf{u}) = \frac{1}{\sqrt{m_x^2 + m_y^2 + 1}} \begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix}, \tag{3.8}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{3.9}$$

$$m_y = \frac{v - c_y}{f_y}. \tag{3.10}$$

In theory, $\Omega = \{\mathbf{p} \in \mathbb{R}^3 \mid z > 0\}$, which limits the field-of-view to less than 180°. In practice, however, this limit is even smaller, typically reducing the allowed field-of-view of the pinhole model to 120°.

Extended Unified Camera Model

The EUCM [33] can be viewed as a generalization of the Unified Camera Model (UCM) [23], both being suitable for fisheye cameras [44]. While the latter projects the 3D point onto the unit sphere and then onto the image plane, in the former the sphere is substituted by an ellipsoid (Figure 3.2).

The EUCM has six intrinsic parameters, namely $\mathbf{c} = (f_x, f_y, c_x, c_y, \alpha, \beta)^\top$, $\alpha \in [0, 1]$, $\beta > 0$, and defines the projection function as:

$$\pi_{\mathbf{c}}(\mathbf{p}) = \begin{pmatrix} f_x \frac{x}{\alpha d + (1-\alpha) z} \\ f_y \frac{y}{\alpha d + (1-\alpha) z} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}, \tag{3.11}$$

$$d = \sqrt{\beta(x^2 + y^2) + z^2}. \tag{3.12}$$

Note that the EUCM degrades to a regular UCM for $\beta = 1$. Moreover, for $\alpha = 0$ the model

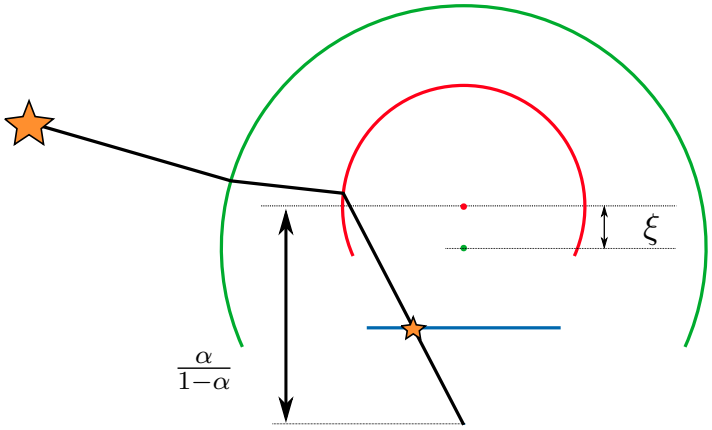


Figure 3.3.: The Double Sphere (**DS**) projection model proposed by Usenko in [61].

degrades to the pinhole model.

The unprojection function of the EUCM is defined as follows:

$$\pi_{\mathbf{c}}^{-1}(\mathbf{u}) = \frac{1}{\sqrt{m_x^2 + m_y^2 + m_z^2}} \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}, \tag{3.13}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{3.14}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{3.15}$$

$$r^2 = m_x^2 + m_y^2, \tag{3.16}$$

$$m_z = \frac{1 - \beta\alpha^2 r^2}{\alpha\sqrt{1 - (2\alpha - 1)\beta r^2} + (1 - \alpha)}, \tag{3.17}$$

Double Sphere Camera Model

In [61] the DSCM is presented, which better fits cameras with fish-eye lenses, has a closed-form inverse, and does not require computationally expensive trigonometric operations, in contrast to the Kannala-Brandt camera model [29]. This model projects a 3D point consecutively onto two concentric unit spheres with centers shifted by ξ. Then, the point is projected onto the image plane using the pinhole model shifted by $\frac{\alpha}{1-\alpha}$ (Figure 3.3). This model has six parameters $\mathbf{c} = (f_x, f_y, c_x, c_y, \xi, \alpha)^\top$ and a projection function defined as follows:

$$\pi_{\mathbf{c}}(\mathbf{p}) = \begin{pmatrix} f_x \frac{x}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \\ f_y \frac{y}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}, \tag{3.18}$$

$$d_1 = \sqrt{x^2 + y^2 + z^2}, \tag{3.19}$$

$$d_2 = \sqrt{x^2 + y^2 + (\xi d_1 + z)^2}. \tag{3.20}$$

To backproject a point on the image plane back to 3D the following equation can be employed:

$$\pi_{\mathbf{c}}^{-1}(\mathbf{u}) = \frac{m_z \xi + \sqrt{m_z^2 + (1 - \xi^2) r^2}}{m_z^2 + r^2} \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \xi \end{pmatrix}, \tag{3.21}$$

$$m_x = \frac{u - c_x}{f_x}, \tag{3.22}$$

$$m_y = \frac{v - c_y}{f_y}, \tag{3.23}$$

$$r^2 = m_x^2 + m_y^2, \tag{3.24}$$

$$m_z = \frac{1 - \alpha^2 r^2}{\alpha \sqrt{1 - (2\alpha - 1) r^2} + 1 - \alpha}. \tag{3.25}$$

3.4. Rigid-body Motion

A rigid-body motion (or rigid-body transformation) is a map

$$g : \mathbb{R}^3 \to \mathbb{R}^3; \quad \mathbf{p} \mapsto g(\mathbf{p}) \tag{3.26}$$

which preserves the distance and orientation between any two points $\mathbf{p}$ and $\mathbf{q}$, i.e.:

$$\|\mathbf{p} - \mathbf{q}\| = \|g(\mathbf{p}) - g(\mathbf{q})\| \qquad \forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3, \tag{3.27}$$

$$g(\mathbf{p}) \times g(\mathbf{q}) = g(\mathbf{p} \times \mathbf{q}) \qquad \forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3. \tag{3.28}$$

The preservation of length and orientation allows us to define the motion g of a rigid body by computing the motion of a Cartesian coordinate frame attached to it. Such a transformation can be decomposed into the motion of the frame's origin by a translation $\mathbf{t} \in \mathbb{R}^3$ and the change in orientation of the frame by a rotation. In total, a rigid-body motion has six degrees of freedom, three accounting for translation and three for rotation.

While several representations exist to express rotation, in this work we define it by a 3×3 orthogonal matrix $\mathbf{R}$ with $\det(\mathbf{R}) = +1$, i.e., an element of the special orthogonal group

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = +1\}. \tag{3.29}$$

Therefore, a complete rigid-body motion can be expressed as a 4×4 transformation matrix $\mathbf{T}$ of the Special Euclidean group $\mathrm{SE}(3)$, which has the form

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.30}$$

and inverse:

$$\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.31}$$

3.4.1. Transformation of a 3D Point

We can define the rigid-body transformation of a point $\mathbf{p} \in \mathbb{R}^3$ with bearing vector $\mathbf{b}$ and inverse distance $d_\mathbf{p}$ as follows:

$$g(\mathbf{p}) \triangleq g(\mathbf{T}, \mathbf{p}) \triangleq g(\mathbf{T}, \mathbf{b}, d_\mathbf{p}) = \mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{3.32}$$

$$= \mathbf{R}\frac{\mathbf{b}}{d_\mathbf{p}} + \mathbf{t}. \tag{3.33}$$

Representing point $\mathbf{p}'$ in homogeneous coordinates, i.e.:

$$\widetilde{\mathbf{p}} \triangleq \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{3.34}$$

allows us to directly use the rigid-body transformation matrix $\mathbf{T}$ to simultaneously apply the rotation and translation onto $\mathbf{p}$, producing the transformed point in homogeneous coordinates

$$\widetilde{\mathbf{p}}' = \mathbf{T}\widetilde{\mathbf{p}} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}\frac{\mathbf{b}}{d_\mathbf{p}} + \mathbf{t} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix}. \tag{3.35}$$

We can therefore introduce the map

$$\widetilde{g}(\widetilde{\mathbf{p}}) \triangleq \widetilde{g}(\mathbf{T}, \widetilde{\mathbf{p}}) \triangleq \widetilde{g}(\mathbf{T}, \mathbf{b}, d_{\mathbf{p}}) = \widetilde{\mathbf{p}}' \tag{3.36}$$

as the rigid-body transformation applied to points in their homogeneous coordinates.

Any number of rigid-body motions can be concatenated by left-multiplying the corresponding transformation matrices. Moreover, the identity transformation, i.e., no motion, is defined by $\mathbf{R} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$, resulting in $g_{identity}(\mathbf{p}) = \mathbf{p}$.

Throughout this thesis, camera poses $T_c \triangleq T_{wc}$ are elements of the Special Euclidean group $\mathrm{SE}(3)$ that transform the coordinates of a point relative to the camera frame $\widetilde{\mathbf{p}}_c = (x, y, z, 1)^\top$ into the world frame, giving

$$\widetilde{\mathbf{p}}_w = \mathbf{T}_{wc}\, \widetilde{\mathbf{p}}_c. \tag{3.37}$$

Inverse Distance Formulation

To be able to deal with 3D points $\mathbf{p}$ at infinity (i.e., with $d_{\mathbf{p}} = 0$) when transforming and projecting them into the image plane we leverage the following insight. All 3D points along the line between the transformed point $\mathbf{p}'$ and the center of projection of the camera project into the same image location. In other words, when scaling point $\mathbf{p}'$ with a constant k it holds true that

$$\pi_{\mathbf{c}}(\mathbf{p}') = \pi_{\mathbf{c}}(k\mathbf{p}'). \tag{3.38}$$

With this in mind, we can scale $\mathbf{p}'$ with $d_{\mathbf{p}}$. By doing so we remove the latter from the denominator in Equation (3.33), thereby producing the scaled point

$$\mathbf{p}'_s = d_{\mathbf{p}}\mathbf{p}' = \mathbf{R}\mathbf{b} + d_{\mathbf{p}}\mathbf{t}, \tag{3.39}$$

where now $d_{\mathbf{p}}$ can safely be zero. Again, note that

$$\pi_{\mathbf{c}}(\mathbf{p}') = \pi_{\mathbf{c}}(\mathbf{p}'_s). \tag{3.40}$$

We can formulate the above transformation using homogeneous coordinates. Note that the homogeneous coordinates of a point can be multiplied by a non-zero constant without modifying the underlying euclidean coordinates. Denoting by $\widetilde{\mathbf{p}}_d$ the homogeneous coordinates of point $\mathbf{p}$ that have been multiplied by $d_{\mathbf{p}}$, and further assuming for now that $d_{\mathbf{p}} \neq 0$, we have that both $\widetilde{\mathbf{p}}$ and $\widetilde{\mathbf{p}}_d$ represent the same 3D point in space, namely

$$\widetilde{\mathbf{p}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \overset{(3.3)}{=} \begin{pmatrix} b_x/d_{\mathbf{p}} \\ b_y/d_{\mathbf{p}} \\ b_z/d_{\mathbf{p}} \\ 1 \end{pmatrix}, \tag{3.41}$$

and

$$\widetilde{\mathbf{p}}_d = d_{\mathbf{p}}\widetilde{\mathbf{p}} = \begin{pmatrix} b_x \\ b_y \\ b_z \\ d_{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ d_{\mathbf{p}} \end{pmatrix}. \tag{3.42}$$

It should be observed that Equation (3.41) is only valid when $d_{\mathbf{p}} \neq 0$, while Equation (3.42) is always valid.

Nevertheless, as long as $d_{\mathbf{p}} \neq 0$, transforming $\widetilde{\mathbf{p}}_d$

$$\widetilde{\mathbf{p}}'_d = \mathbf{T}\widetilde{\mathbf{p}}_d = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ d_{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} \mathbf{R}\mathbf{b} + d_{\mathbf{p}}\mathbf{t} \\ d_{\mathbf{p}} \end{pmatrix} \overset{(3.39)}{=} \begin{pmatrix} \mathbf{p}'_s \\ d_{\mathbf{p}} \end{pmatrix} \tag{3.43}$$

is equivalent to transforming $\widetilde{\mathbf{p}}$ (cf. Equation (3.35)). Indeed, if we normalize $\widetilde{\mathbf{p}}'_d$ by dividing each of its components by the last one (i.e., by $d_{\mathbf{p}}$), and still assuming that $d_{\mathbf{p}} \neq 0$, we obtain $\widetilde{\mathbf{p}}'$ back as defined in Equation (3.35):

$$\widetilde{\mathbf{p}}' = \frac{1}{d_{\mathbf{p}}} \begin{pmatrix} \mathbf{p}'_s \\ d_{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} \dfrac{\mathbf{p}'_s}{d_{\mathbf{p}}} \\ 1 \end{pmatrix} \overset{(3.39)}{=} \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix}. \tag{3.44}$$

Since Equation (3.44) is only valid when $d_{\mathbf{p}} \neq 0$, in this work we employ the formulation given by Equation (3.43), which effectively allows to transform any point, including points at infinity, given that $d_{\mathbf{p}}$ can safely be zero or close to zero.

For completeness, we can also introduce the map

$$\widetilde{g}_d(\widetilde{\mathbf{p}}_d) \triangleq \widetilde{g}_d(\mathbf{T}, \widetilde{\mathbf{p}}_d) \triangleq \widetilde{g}_d(\mathbf{T}, \mathbf{b}, d_{\mathbf{p}}) = \widetilde{\mathbf{p}}'_d \tag{3.45}$$

as the rigid-body transformation applied to points in their unnormalized homogeneous coordinates (cf. Equation (3.42)). We can also extend the projection of 3D points into the image plane as defined in (3.4) to also accept a point in its homogeneous coordinates (both in its normalized, i.e., last component 1, or unnormalized version). To this end, we define a 3×4 matrix $\mathbf{M}$ as

$$\mathbf{M} = \begin{pmatrix} \mathbf{I}_{3\times 3} & \mathbf{0} \end{pmatrix} \tag{3.46}$$

that extracts the first three components from the homogeneous representation of a point. Thus, projecting for instance $\widetilde{\mathbf{p}}'$ (normalized homogeneous coordinates) and $\widetilde{\mathbf{p}}'_d$ (unnormalized homogeneous coordinates) produces, respectively,

$$\pi_{\mathbf{c}}(\widetilde{\mathbf{p}}') \triangleq \pi_{\mathbf{c}}(\mathbf{M}\widetilde{\mathbf{p}}') \overset{(3.44)}{=} \pi_{\mathbf{c}}(\mathbf{p}'), \tag{3.47}$$

$$\pi_{\mathbf{c}}(\widetilde{\mathbf{p}}'_d) \triangleq \pi_{\mathbf{c}}(\mathbf{M}\widetilde{\mathbf{p}}'_d) \overset{(3.43)}{=} \pi_{\mathbf{c}}(\mathbf{p}'_s), \tag{3.48}$$

where both projections are equivalent, as discussed above (cf. Equation (3.40)).

Finally, it can be of interest to compute the inverse distance $d_{\mathbf{p}'}$ of the transformed point $\mathbf{p}'$ as a function of the inverse distance $d_{\mathbf{p}}$ of the original point $\mathbf{p}$, which can be easily derived as follows:

$$d_{\mathbf{p}'} \triangleq \frac{1}{\|\mathbf{p}'\|} \overset{(3.39)}{=} \frac{1}{\left\|\frac{\mathbf{p}'_s}{d_{\mathbf{p}}}\right\|} = \frac{d_{\mathbf{p}}}{\|\mathbf{p}'_s\|}. \tag{3.49}$$

3.4.2. Lie Algebra

While the representation of translation as a vector $\mathbf{t}$ is canonical –the three components of $\mathbf{t}$ correspond to the three degrees of freedom–, this is not true when representing a rotation with a rotation matrix $\mathbf{R}$, since it requires nine parameters for only three degrees of freedom. (Note that enforcing orthogonality ($\mathbf{R}^\top\mathbf{R} = \mathbf{I}$) and $\det(\mathbf{R}) = +1$ imposes six constraints and only leaves three free parameters.)

To obtain a minimal representation of a rigid-body transformation $G \in \mathrm{SE}(3)$ (and, in general, of an element in any Lie group) we can use the parameters of the associated Lie algebra, which for the case of $\mathrm{SE}(3)$ is denoted as $\mathrm{se}(3)$.

Exponential Map

For any Lie group the tangent space at the identity is the corresponding Lie algebra. In particular, for $\mathrm{SE}(3)$ any transformation matrix $\mathbf{T} \in \mathrm{SE}(3)$ has a representation in its Lie algebra $\mathrm{se}(3)$ through a 6×1 velocity vector $\boldsymbol{\xi}$ known as the twist coordinates, which we denote as

$$\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{pmatrix}, \tag{3.50}$$

where $\boldsymbol{\nu} = (\nu_1, \nu_2, \nu_3)^\top$ is the linear velocity (related to translation) and $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^\top$ the angular velocity (related to rotation). The mapping from the Lie algebra to the Lie group is given by the exponential map (or matrix exponential):

$$\exp : \mathrm{se}(3) \to \mathrm{SE}(3); \quad \widehat{\boldsymbol{\xi}} \mapsto \mathrm{e}^{\widehat{\boldsymbol{\xi}}}, \tag{3.51}$$

$$\mathbf{T} = \mathrm{e}^{\widehat{\boldsymbol{\xi}}}, \tag{3.52}$$

where $\widehat{\boldsymbol{\xi}} \in \mathrm{se}(3)$ is a 4×4 matrix known as the twist, defined as

$$\widehat{\boldsymbol{\xi}} \triangleq \begin{pmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{pmatrix}^\wedge \triangleq \begin{pmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{\nu} \\ \mathbf{0} & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 & \nu_1 \\ \omega_3 & 0 & -\omega_1 & \nu_2 \\ -\omega_2 & \omega_1 & 0 & \nu_3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4}. \tag{3.53}$$

Again, note that the set of all twists form the tangent space at the identity, which is precisely the Lie algebra, and the matrix exponential maps an element from the tangent space to the corresponding matrix Lie group.

Typically, applying the *hat*-operator to a vector $\mathbf{u} \in \mathbb{R}^3$ creates a skew-symmetric matrix $\widehat{\mathbf{u}}$ (sometimes denoted as $[\mathbf{u}]_\times$) of the form

$$\widehat{\mathbf{u}} \triangleq [\mathbf{u}]_\times = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \in \mathbb{R}^{3\times 3}. \tag{3.54}$$

With a slight abuse of notation, when applied to the twist coordinates $\boldsymbol{\xi} \in \mathbb{R}^6$ in Equation (3.53), the $\widehat{\cdot}$-operator creates a 4×4 matrix with the 3×3 skew-symmetric matrix $\widehat{\boldsymbol{\omega}}$ in the upper left and the translational velocity parameters $\boldsymbol{\nu}$ in the last column (plus a row of zeros at the last row).

The exponential map in Equation (3.52) has a closed form solution [41] given by

$$\mathrm{e}^{\widehat{\boldsymbol{\xi}}} = \begin{pmatrix} \mathrm{e}^{\widehat{\boldsymbol{\omega}}} & \mathbf{V}\boldsymbol{\nu} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{3.55}$$

where $e^{\widehat{\boldsymbol{\omega}}}$ can be obtained through Rodrigues' formula through

$$\mathrm{e}^{\widehat{\boldsymbol{\omega}}} = \mathbf{I} + \frac{\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|}\widehat{\boldsymbol{\omega}} + \frac{1 - \cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2}\widehat{\boldsymbol{\omega}}^2, \tag{3.56}$$

and $\mathbf{V}$ is defined as

$$\mathbf{V} = \mathbf{I} + \frac{1 - \cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2}\widehat{\boldsymbol{\omega}} + \frac{\|\boldsymbol{\omega}\| - \sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^3}\widehat{\boldsymbol{\omega}}^2. \tag{3.57}$$

The exponential map has similar properties to the exponential function, including

$$\frac{\partial \exp(t\,\widehat{\boldsymbol{\xi}})}{\partial t} = \widehat{\boldsymbol{\xi}}\exp(t\,\widehat{\boldsymbol{\xi}}) = \exp(t\,\widehat{\boldsymbol{\xi}})\widehat{\boldsymbol{\xi}}, \tag{3.58}$$

and

$$\text{if} \quad \widehat{\boldsymbol{\xi}}_1\widehat{\boldsymbol{\xi}}_2 = \widehat{\boldsymbol{\xi}}_2\widehat{\boldsymbol{\xi}}_1 \implies \exp(\widehat{\boldsymbol{\xi}}_1)\exp(\widehat{\boldsymbol{\xi}}_2) = \exp(\widehat{\boldsymbol{\xi}}_1 + \widehat{\boldsymbol{\xi}}_2). \tag{3.59}$$

Given a rigid-body transformation $\mathbf{A} \in \mathrm{SE}(3)$, it can also be shown that

$$\mathbf{A}\exp(\widehat{\boldsymbol{\xi}})\mathbf{A}^{-1} = \exp(\mathbf{A}\widehat{\boldsymbol{\xi}}\mathbf{A}^{-1}). \tag{3.60}$$

We refer the reader to [51] (2002, pp.2) for proof and to Strasdat's PhD thesis [54] for a concise summary of Lie Groups.

Logarithmic Map

The inverse mapping from the Lie group to the Lie algebra is called the logarithmic map, and for the Special Euclidean group $\mathrm{SE}(3)$ it is defined as follows:

$$\log : \mathrm{SE}(3) \to \mathrm{se}(3); \quad \log(\mathrm{e}^{\widehat{\boldsymbol{\xi}}}) \mapsto \widehat{\boldsymbol{\xi}}, \tag{3.61}$$

$$\widehat{\boldsymbol{\xi}} = \log(\mathbf{T}). \tag{3.62}$$

Adjoint Map

Let $\mathbf{A} \in \mathrm{SE}(3)$, the adjoint representation of $\mathrm{SE}(3)$ is a map

$$\mathrm{Adj}_{\mathbf{A}} : \mathrm{se}(3) \to \mathrm{se}(3); \quad \mathrm{Adj}_{\mathbf{A}}(\widehat{\boldsymbol{\xi}}) \triangleq \mathbf{A}\widehat{\boldsymbol{\xi}}\mathbf{A}^{-1}. \tag{3.63}$$

Since function $\mathrm{Adj}(\cdot)$ is a linear operator, there exists a matrix $\mathrm{Ad}(\mathbf{A}) \in \mathbb{R}^{4\times 4}$ such that

$$\widehat{\mathrm{Ad}(\mathbf{A}) \cdot \boldsymbol{\xi}} = \mathbf{A}\widehat{\boldsymbol{\xi}}\mathbf{A}^{-1} (= \mathrm{Adj}_{\mathbf{A}}(\widehat{\boldsymbol{\xi}})). \tag{3.64}$$

In particular, the adjoint map of $\mathrm{SE}(3)$ is

$$\mathrm{Ad}(\mathbf{A}) = \begin{pmatrix} \mathbf{R} & \widehat{\mathbf{t}\mathbf{R}} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}. \tag{3.65}$$

Applying property (3.60) to the above definition we obtain

$$\exp(\widehat{\mathrm{Ad}(\mathbf{A}) \cdot \boldsymbol{\xi}}) = \mathbf{A} \exp(\widehat{\boldsymbol{\xi}})\mathbf{A}^{-1}, \tag{3.66}$$

thus allowing us to move the exponential map from the right-hand side to the left-hand side of $\mathbf{A}$ [54], i.e.:

$$\mathbf{A} \cdot \exp(\widehat{\boldsymbol{\xi}}) = \exp(\widehat{\mathrm{Ad}(\mathbf{A}) \cdot \boldsymbol{\xi}}) \cdot \mathbf{A}. \tag{3.67}$$

3.5. Least Squares

3.5.1. General Approach

The method of least squares is a standard approach in regression analysis to approximate the solution of overdetermined systems, where an exact solution is typically not available due to noise or a too simplified model. The goal is to estimate the parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_m)^\top$ of a model $\boldsymbol{f}(\boldsymbol{\theta})$ that minimize the sum of squared errors (or Sum over Squared Distances (SSD)) between n observations $\hat{\boldsymbol{f}} = (\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n)^\top$ and the corresponding model's values $\boldsymbol{f}(\boldsymbol{\theta}) = (f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta}), \ldots, f_n(\boldsymbol{\theta}))^\top$, i.e.:

$$E_{\mathrm{LS}}(\boldsymbol{\theta}) = (\boldsymbol{f}(\boldsymbol{\theta}) - \hat{\boldsymbol{f}})^\top (\boldsymbol{f}(\boldsymbol{\theta}) - \hat{\boldsymbol{f}}) = \sum_i^n (f_i(\boldsymbol{\theta}) - \hat{f}_i)^2 = \sum_i^n (r_i(\boldsymbol{\theta}))^2, \tag{3.68}$$

where $r_i(\boldsymbol{\theta}) = f_i(\boldsymbol{\theta}) - \hat{f}_i$ is the i^{th} residual. The best parameters $\boldsymbol{\theta}_{\mathrm{LS}}$ are those that minimize the total error E_{LS}:

$$\boldsymbol{\theta}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \; E_{\mathrm{LS}}(\boldsymbol{\theta}). \tag{3.69}$$

We can calculate the partial derivatives of E_{LS} with respect to the parameters $\boldsymbol{\theta}$ and set them equal to zero, obtaining

$$\frac{\partial E_{\mathrm{LS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i}^{n} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} 2 r_i(\boldsymbol{\theta}) = \mathbf{0}, \tag{3.70}$$

where solving for $\boldsymbol{\theta}$ provides the least squares solution $\boldsymbol{\theta}_{\mathrm{LS}}$. Note that a unique solution may only exist if the number of observations n is greater or equal than the number of parameters m in the model. This is not a problem in practice, since typically we encounter largely overconstrained systems with many more observations than parameters.

Linear Least Squares

If $f_i(\boldsymbol{\theta})$ is linear in its parameters $\boldsymbol{\theta}$, i.e.:

$$f_{\mathrm{lin}}(\boldsymbol{\theta}) = \mathbf{a}_i^\top \boldsymbol{\theta}, \tag{3.71}$$

with $\mathbf{a}_i \in \mathbb{R}^m$, then Equation (3.69) is a linear least squares problem whose solution can be obtained in closed form as follows. By first plugging $f_{\mathrm{lin}}(\boldsymbol{\theta})$ into Equation (3.70)

$$\sum_{i}^{n} \mathbf{a}_i^\top \left(\mathbf{a}_i^\top \boldsymbol{\theta} - \hat{f}_i \right) = \mathbf{0}, \tag{3.72}$$

then transposing the whole equation and applying the matrix property $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$, we obtain

$$\sum_{i}^{n} \mathbf{a}_i \left(\mathbf{a}_i^\top \boldsymbol{\theta} - \hat{f}_i \right) = \mathbf{0}. \tag{3.73}$$

Rearranging terms we can write the above equation as

$$\sum_{i}^{n} \mathbf{a}_i \mathbf{a}_i^\top \boldsymbol{\theta} = \sum_{i}^{n} \hat{f}_i \mathbf{a}_i. \tag{3.74}$$

Finally, stacking all n vectors $\mathbf{a}_i \in \mathbb{R}^m$ as row vectors into a matrix

$$\mathbf{A} = \begin{pmatrix} — & \mathbf{a}_1^\top & — \\ — & \mathbf{a}_2^\top & — \\ & \vdots & \\ — & \mathbf{a}_n^\top & — \end{pmatrix} \in \mathbb{R}^{n \times m}, \tag{3.75}$$

we can write the so-called normal equations

$$\mathbf{A}^\top \mathbf{A}\, \boldsymbol{\theta} = \mathbf{A}^\top \hat{\boldsymbol{f}}, \tag{3.76}$$

whose solution is the parameter vector $\boldsymbol{\theta}$.

Non-Linear Least Squares

When $f_i(\boldsymbol{\theta})$ has a non-linear dependence on $\boldsymbol{\theta}$ the least squares method is referred to as non-linear least squares. The procedure in this case if to first obtain a linear dependence on the model parameters $\boldsymbol{\theta}$ by linearizing the residuals $r_i(\boldsymbol{\theta})$ using a first order Taylor expansion at $\boldsymbol{\theta} = \boldsymbol{\alpha}$:

$$r_{\text{lin},i}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\alpha}} = r_i(\boldsymbol{\alpha}) + \left.\frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\alpha}} (\boldsymbol{\theta} - \boldsymbol{\alpha}) = r_i(\boldsymbol{\alpha}) + \mathbf{J}_i(\boldsymbol{\alpha})(\boldsymbol{\theta} - \boldsymbol{\alpha}), \tag{3.77}$$

where the row vector $\mathbf{J}_i(\boldsymbol{\alpha}) \in \mathbb{R}^{1 \times m}$ is the Jacobian of the i^{th} residual evaluated at $\boldsymbol{\theta} = \boldsymbol{\alpha}$. Since Equation (3.77) is an approximation of $r_i(\boldsymbol{\theta})$ at $\boldsymbol{\theta} = \boldsymbol{\alpha}$, the solution has to be obtained iteratively. At every iteration $k+1$ the residual is relinearized around the last solution $\boldsymbol{\theta}_k$, giving

$$r_{\text{lin},i}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = r_i(\boldsymbol{\theta}_k) + \mathbf{J}_i(\boldsymbol{\theta}_k)(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) = r_i(\boldsymbol{\theta}_k) + \mathbf{J}_i(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta}, \tag{3.78}$$

where $\Delta\boldsymbol{\theta}$ is the increment that needs to be computed and later added to the previous solution, i.e.:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}. \tag{3.79}$$

To compute $\Delta\boldsymbol{\theta}$ we can plug Equation (3.78) into (3.70), obtaining

$$\sum_i^n \mathbf{J}_i(\boldsymbol{\theta}_k)^\top \Big(r_i(\boldsymbol{\theta}_k) + \mathbf{J}_i(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta} \Big) = \mathbf{0}, \tag{3.80}$$

$$\sum_i^n \mathbf{J}_i(\boldsymbol{\theta}_k)^\top \mathbf{J}_i(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta} = -\sum_i^n \mathbf{J}_i(\boldsymbol{\theta}_k)^\top r_i(\boldsymbol{\theta}_k). \tag{3.81}$$

Rewriting the above equations into matrix notation gives

$$\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{J}(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta} = -\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{r}(\boldsymbol{\theta}_k), \tag{3.82}$$

where $\mathbf{J}(\boldsymbol{\theta}_k) \in \mathbb{R}^{n\times m}$ is the Jacobian matrix with $\mathbf{J}_i(\boldsymbol{\theta}_k) \in \mathbb{R}^{1\times m}$ stacked as row vectors; $\mathbf{r}(\boldsymbol{\theta}_k) \in \mathbb{R}^{n\times 1}$ is the stacked vector of all residuals.

This procedure –iteratively relinearizing the residual term at the last estimate and solving the resulting normal equations– is known as the Gauss-Newton (G-N) method. Convergence is not ensured; in fact, G-N does not even guarantee a descent in the error. For this reason, the solution is highly influenced by the initial estimate $\boldsymbol{\theta}_0$, which should be therefore close to the true solution. One extension to the G-N method –among others– is the Levenberg-Marquardt (L-M) algorithm:

$$\left(\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{J}(\boldsymbol{\theta}_k) + \lambda \operatorname{diag}\left(\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{J}(\boldsymbol{\theta}_k)\right)\right)\Delta\boldsymbol{\theta} = -\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{r}(\boldsymbol{\theta}_k), \tag{3.83}$$

which behaves like a gradient descent method when the parameters are far from their optimal value (by means of increasing a dampening factor λ), and like the G-N method when the parameters are close to their optimal value (small λ). Such an optimization procedure ensures that at every step the cost is reduced, but also encourages super-linear convergence close to a local minimum.

3.5.2. Bayesian Derivation

The least squares approach can also be derived from a Bayesian perspective by maximizing the a posteriori likelihood of the parameters $\boldsymbol{\theta}$ given the observations $\hat{\boldsymbol{f}}$:

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \hat{\boldsymbol{f}}). \tag{3.84}$$

Applying Bayes's rule we obtain

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \frac{p(\hat{\boldsymbol{f}} \mid \boldsymbol{\theta})\; p(\boldsymbol{\theta})}{p(\hat{\boldsymbol{f}})}, \tag{3.85}$$

where we can drop $p(\hat{\boldsymbol{f}})$ as it does not depend on $\boldsymbol{\theta}$, giving

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\hat{\boldsymbol{f}} \mid \boldsymbol{\theta})\; p(\boldsymbol{\theta}). \tag{3.86}$$

Assuming a uniform distribution for $p(\boldsymbol{\theta})$, i.e., assigning a constant probability to all possible parameters (essentially), we can also drop $p(\boldsymbol{\theta})$, thus obtaining

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\hat{\boldsymbol{f}} \mid \boldsymbol{\theta}). \tag{3.87}$$

By further assuming that all observations $\hat{\boldsymbol{f}}$ are independent and identically distributed (i.i.d.) we get

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \prod_{i}^{n} p(\hat{f}_i \mid \boldsymbol{\theta}). \tag{3.88}$$

Minimizing instead the negative log-likelihood can help us simplify computation later on:

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta}} \sum_{i}^{n} -\log(p(\hat{f}_i \mid \boldsymbol{\theta})). \tag{3.89}$$

When $p(\hat{f}_i \mid \boldsymbol{\theta})$ follows a normal distribution $\mathcal{N}(\hat{f}_i, \mu, \sigma)$ with $\mu = f(x_i, \boldsymbol{\theta})$ and standard deviation σ, Equation (3.89) amounts to

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta}} \sum_{i}^{n} \frac{1}{2\sigma^2}(\hat{f}_i - f(x_i, \boldsymbol{\theta}))^2 = \arg\min_{\boldsymbol{\theta}} \sum_{i}^{n} \frac{1}{2\sigma^2}(r_i(\boldsymbol{\theta}))^2, \tag{3.90}$$

which is equivalent to Equation (3.69), since $\frac{1}{2\sigma^2}$ is constant and can be dropped. Note that we can derive the same formulation by assuming $p(r_i \mid \boldsymbol{\theta}) = \mathcal{N}(r_i(\boldsymbol{\theta}), 0, \sigma)$ instead.

3.5.3. Weighted Least Squares

Due to the quadratic term in Equation (3.68), outliers, i.e., observations that produce large residuals $r_i(\boldsymbol{\theta})$ and that cannot be explained by the model, have a heavy (negative) influence on the estimated parameters. The influence of these outlier observations can be mitigated by replacing the quadratic term x^2 with a robust error function $\rho(x)$, thus rewriting Equation (3.69) as

$$\boldsymbol{\theta}_{\text{WLS}} = \arg\min_{\boldsymbol{\theta}} \; E_{\text{WLS}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i}^{n} \rho(r_i(\boldsymbol{\theta})). \tag{3.91}$$

The weighted least squares solution $\boldsymbol{\theta}_{\text{WLS}}$ can be found by computing the partial derivatives of Equation (3.91) with respect to $\boldsymbol{\theta}$ and equating it to zero, i.e.:

$$\frac{\partial E_{\mathrm{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i}^{n} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \psi(r_i(\boldsymbol{\theta})) = \mathbf{0}, \tag{3.92}$$

where $\psi(x)$ is defined as the influence function and is the derivative of the robust error function $\rho(x)$, i.e., $\psi(x) = \rho'(x)$. Bearing in mind that

$$w(x) = \frac{\psi(x)}{x} = \frac{\rho'(x)}{x}, \tag{3.93}$$

we can derive the alternative formulation

$$\frac{\partial E_{\mathrm{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i}^{n} \frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} w(r_i(\boldsymbol{\theta})) r_i(\boldsymbol{\theta}) = \mathbf{0}. \tag{3.94}$$

Note that Equation (3.94) also minimizes the Iteratively Reweighted Least Squares (IRLS) problem [58]

$$E_{\mathrm{IRLS}}(\boldsymbol{\theta}) = \sum_{i}^{n} w(r_i(\boldsymbol{\theta})) \big(r_i(\boldsymbol{\theta})\big)^2, \tag{3.95}$$

where each weight $w_i(r_i(\boldsymbol{\theta}))$ scaling its corresponding residual $r_i(\boldsymbol{\theta})$ will be treated as constant during every iteration. (This is an approximation, but allows us obtain linear normal equations.) We can incorporate the weights into the normal equations (cf. Equation (3.82)) by means of a diagonal $n \times n$ matrix $\mathbf{W} = \mathrm{diag}(w_1, w_2, \ldots, w_n)$, finally obtaining

$$\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{W} \mathbf{J}(\boldsymbol{\theta}_k) \Delta\boldsymbol{\theta} = -\mathbf{J}(\boldsymbol{\theta}_k)^\top \mathbf{W} \mathbf{r}(\boldsymbol{\theta}_k), \tag{3.96}$$

where both the weights and the new parameter estimates need to be recomputed until convergence.

Robust Scale Estimator

If residuals follow a normal distribution, i.e., $p(r_i \mid \boldsymbol{\theta}) = \mathcal{N}(r_i(\boldsymbol{\theta}), \mathbf{0}, \sigma)$, then the robust error function is $\rho(x) = \frac{1}{2}(\frac{x}{\sigma})^2$, the influence function $\psi(x) = \frac{x}{\sigma^2}$, and the weight function results in $w(x) = \frac{1}{\sigma^2}$. However, when the residuals follow a distribution $p(r_i \mid \boldsymbol{\theta})$ such that the scale parameter σ cannot be factored out of the weight function (in contrast to the normal distribution), then σ must be estimated as well.

Since outliers have a large influence on the scale σ, typically robust methods are employed in

order to obtain a reliable estimate. A very common robust estimator for the scale is obtained through the use of the Median Absolute Deviation (MAD), which provides a robust scale estimate that we denote $\hat{\sigma}$, and obtained as follows:

$$\hat{\sigma}(\mathbf{x}) = c \cdot \mathrm{MAD} \tag{3.97}$$
$$= c \cdot \mathrm{median}(|\mathbf{x} - \mathrm{median}(\mathbf{x})|), \tag{3.98}$$

where $c = 1.4826$ is a constant scale factor if the data is normally distributed.

3.6. Miscellaneous

3.6.1. Coupled and Decoupled Increments

Incrementing an element $\mathbf{T} \in \mathrm{SE}(3)$ with an increment $\Delta\mathbf{T}$ with the form

$$\Delta\mathbf{T} = \begin{pmatrix} \Delta\mathbf{R} & \Delta\mathbf{t} \\ 0 & 1 \end{pmatrix} \tag{3.99}$$

can be achieved either in a coupled or a decoupled manner. Moreover, the increment can be added to the element either as a left or right increment. Throughout this work we use the left-incremental approach, for which a coupled increment would result in

$$\mathbf{T} \boxplus \Delta\mathbf{T} \triangleq \begin{pmatrix} \Delta\mathbf{R} & \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \triangleq \begin{pmatrix} \Delta\mathbf{R}\,\mathbf{R} & \Delta\mathbf{R}\,\mathbf{t} + \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{3.100}$$

which essentially amounts to multiplying the two elements of group. A decoupled approach, on the other hand, would increment the current $\mathbf{T}$ as follows:

$$\mathbf{T} \boxplus \Delta\mathbf{T} \triangleq \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \boxplus \begin{pmatrix} \Delta\mathbf{R} & \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \triangleq \begin{pmatrix} \Delta\mathbf{R}\,\mathbf{R} & \mathbf{t} + \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.101}$$

The decoupled increment may be preferred in general, and specially when $\mathbf{t}$ is large, since a small increment in rotation $\Delta\mathbf{R}$ could cause too big of an increase in the translational component of the resulting $\mathrm{SE}(3)$ element. Moreover, this decoupled increment is less computationally expensive than its coupled counterpart.

Furthermore, obtaining the increment $\Delta\mathbf{T}$ can as well be obtained in a coupled or decoupled manner. The former corresponds to applying the exponential map as defined in (3.55) to the increment in its twist form $\widehat{\boldsymbol{\xi}}$ as defined in (3.53), thus giving

$$\Delta\mathbf{T} = \exp(\widehat{\boldsymbol{\xi}}) = \exp\left(\begin{pmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{pmatrix}^{\wedge}\right). \tag{3.102}$$

The decoupled alternative, on its part, generates the increment $\Delta\mathbf{T}$ by computing a rotational increment $\Delta\mathbf{R}$ from the angular velocity $\boldsymbol{\omega}$, i.e.:

$$\Delta\mathbf{R} = \exp(\widehat{\boldsymbol{\omega}}) \ (= \mathrm{e}^{\widehat{\boldsymbol{\omega}}}), \tag{3.103}$$

and simply uses the linear velocity $\boldsymbol{\nu}$ as the translational increment $\Delta\mathbf{t}$, resulting in

$$\Delta\mathbf{T} = \begin{pmatrix} \exp(\widehat{\boldsymbol{\omega}}) & \boldsymbol{\nu} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.104}$$

Computing $\Delta\mathbf{T}$ through the decoupled approach (Equation (3.104)) is less computationally expensive than employing Equation (3.102), since in the former we do not compute $\mathbf{V}$ (cf. Equations (3.55) and (3.57)).

4. Approach

The system we present in this work jointly optimizes structure (map points) and motion (camera poses) by minimizing the photometric error. In contrast to [13] we do not follow a sliding window plus marginalization scheme, but rather rely on both temporal and covisibility constraints, thus being able to reuse old map points when possible, as proposed in [64]. This strategy allows us to build a consistent map with few duplicate map points, as we describe in Section 4.1.

As is common in most SLAM systems [35, 47, 13], our method consists of a tracking front-end and an optimization back-end, which can run in two parallel threads:

1. The tracking thread uses DIA to track a new frame against a local map at frame rate, while also marking tracked frames as keyframes when necessary. The front-end of the system is covered in Chapter 5.

2. The mapping thread uses newly tracked frames to trace candidate points from active keyframes similarly to [15]. Furthermore, if the new frame was marked as keyframe by the tracking thread, we recompute the local window of active keyframes, possibly activating candidate points (i.e., upgrading them to landmarks) and running PBA to jointly optimize over the newly active window of keyframes and their hosted landmarks. Outlier removal, occlusions detection and point deduplication are also taken care of in the back-end, which we present in Chapter 6.

We use a relative point formulation (where landmarks in the map are relative to their host frame) plus an inverse distance parametrization (cf. Section 3.2). Additionally, we keep the "direction" of a point fixed (cf. $\mathbf{b}$ in Equation (3.1)) and only optimize one Degree of Freedom (DoF) (i.e., the inverse distance), as employed by other direct approaches [13, 64, 42].

Note that camera intrinsic parameters are estimated in advance through calibration and are not further refined during operation.

Figure 4.1 presents and overview of the system. The key aspect to it is the Local Covisibility Window of active keyframes, which we describe next.

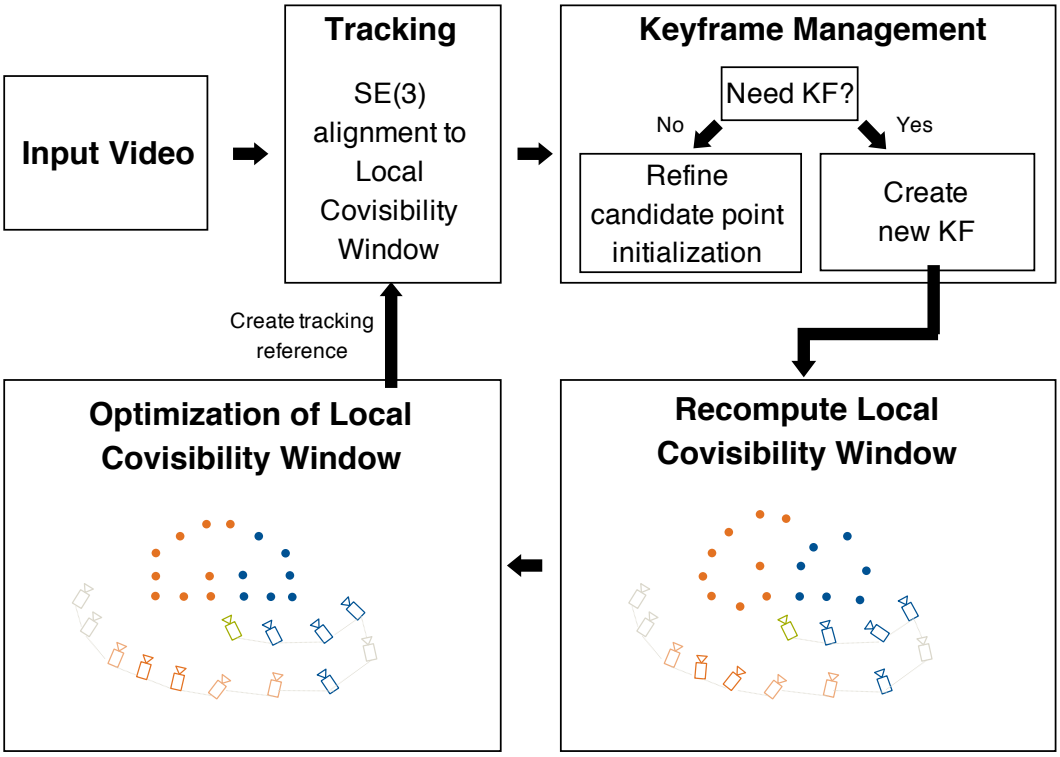


Figure 4.1.: Overview of the visual SLAM system presented in this work.

4.1. Covisibility Window

We follow the strategy presented in [64] by Zubizarreta et al. and leverage a combination of temporal and covisible keyframes to form a Local Covisibility Window (LCW) of active keyframes on which to run PBA, where by covisible keyframes we refer to keyframes that observed landmarks hosted in any of the temporal keyframes. The formation process of the LCW is schematized in Figure 4.2.

The temporal part is a sliding window of temporally connected keyframes as in [13], which is responsible of creating new map points and maintaining accuracy during exploration. When a new keyframe is required based on the criteria presented in Section 5.2, it is inserted into the temporal window (I_0 in the third stage of Figure 4.2). Moreover, to maintain a fixed number of temporal keyframes N_t (4 in our experiments) we remove an old keyframe from the temporal part according to the criteria presented in Section 5.3.

To build the covisibility part of the LCW we project the landmarks hosted in the temporal keyframes into every other *old* keyframe (i.e., all the keyframes that are not part of the temporal window). We will select a maximum of N_c covisible keyframes (15 in our experiments), those that observe most landmarks hosted in the temporal window. Out of these N_c covisible keyframes we aim to activate a maximum of N_{ac} *active covisible* keyframes (3 in our experiments), whose poses and hosted landmarks will be optimized during PBA. As described in [64], the intuition behind these active

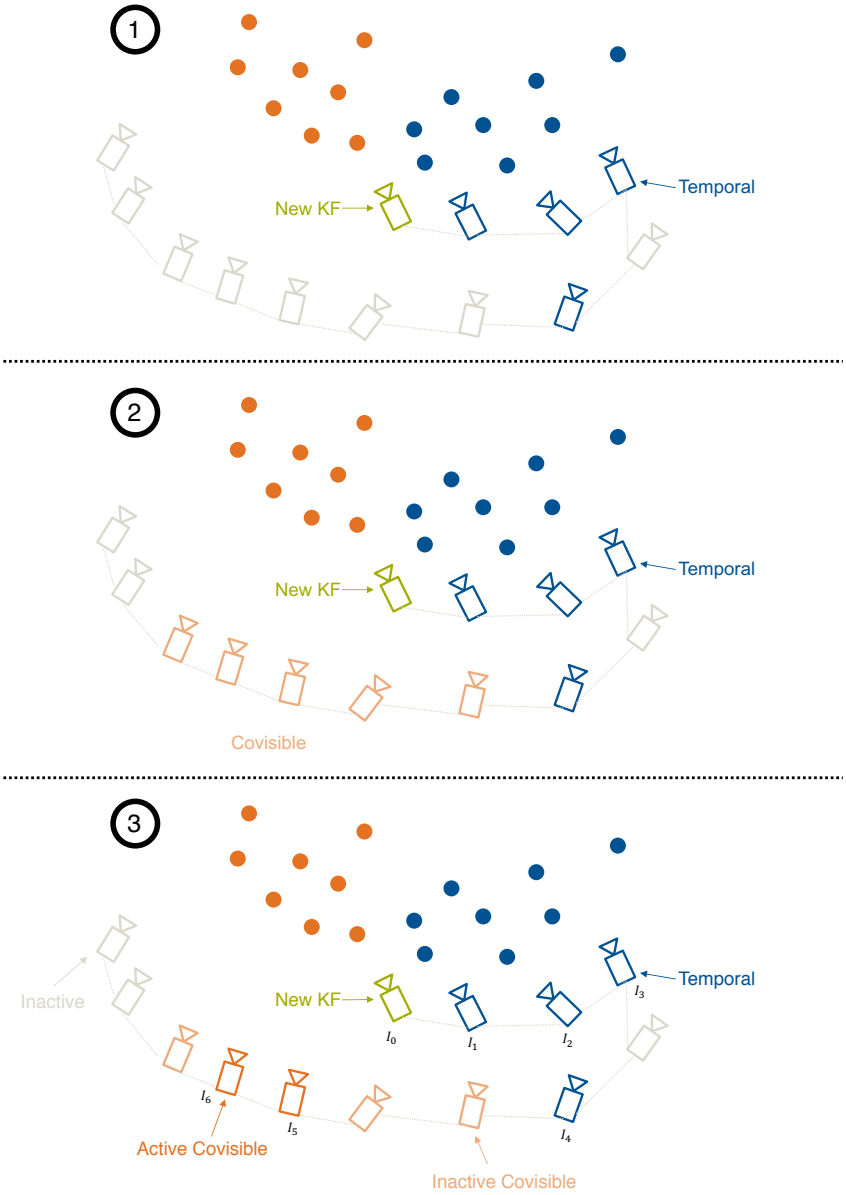


Figure 4.2.: Computation of the LCW, based on that presented in [64]. When a new keyframe is created, (1), we project all landmarks hosted in the current temporal window of N_t keyframes (blue cameras) into old keyframes (gray cameras in (1)). We then select those old keyframes that share the most observations with the temporal window, resulting in an initial set of N_c covisible keyframes (light orange cameras in (2)). Finally, in (3) we project the landmarks hosted in this initial list of N_c covisible keyframes into the new keyframe (green camera), and select a maximum number of N_{ac} active covisible keyframes based on how many of their hosted landmarks fall into depleted areas of a distance map that we build for the last keyframe. Active covisible and temporal keyframes (and their hosted landmarks) will be included in the local window PBA. Covisible keyframes that are not activated are kept as inactive covisible keyframes, and will contribute observations of active landmarks to the PBA.

covisible keyframes is to incorporate already mapped areas into the local optimization before activating new map points, thus avoiding map point duplication and ensuring map consistency. To this end, active covisible keyframes are selected based on how many of their hosted points fall into depleted areas of a distance map that we build for the newly created temporal keyframe (I_0). This distance map registers, for every pixel, the distance Manhattan distance to the closest map point projection.

The remaining $N_{ic} = N_c - N_{ac}$ covisible keyframes will remain *inactive* during PBA, meaning their poses and hosted landmarks will remain fixed during PBA. Nevertheless, they will be part of the optimization in that they will contribute observations of active landmarks to the optimization, thus helping fix the scale and pose gauge.

Next, we describe the system's front-end in Chapter 5, while in Chapter 6 we present the back-end of the system, delving into point management (Section 6.1), PBA (Section 6.2) and outlier management (Section 6.3).

5. Front-End

5.1. Initial Frame Tracking

Every new frame is initially tracked with respect to a tracking reference using conventional two-frame DIA, where the tracking reference is the result of projecting all active landmarks into the latest keyframe, where by active landmarks we refer to those hosted in either temporal or active covisible keyframes (cf. Section 4.1). We use a multi-scale image pyramid approach to deal with large displacements. Moreover, we have explored the use of a constant velocity model versus simply initializing the pose of the new frame with that of the previous frame. We have found the latter to be a better choice, as also concluded in [30].

As in [52] and [13], when down-scaling the inverse distance map of the tracking reference (the latest keyframe), a pixel is assigned an inverse distance value if at least one of the source pixels in the finer level has a valid inverse distance value. When there are more than one valid values, the pixel in the coarser level will be assigned the largest inverse distance, which corresponds to the closest point to the camera, thus helping to avoid occlusions early on.

Given the intensity images of a host frame I_h (i.e., the last keyframe) and a target frame I_t (i.e., the new frame to be tracked), as well as the inverse distance map of the host frame, DIA consists in computing the camera motion that transforms both images into a common viewpoint where the intensity differences between both intensity images is zero for all pixels. This definition is only valid under the photo-consistency assumption, where all viewed surfaces are assumed to be Lambertian, i.e., they have the same radiance when viewed from any angle. In practice, the summed intensity difference, or error, over all pixels is never zero due to occlusions, moving objects and sensor noise. However, we can still estimate the unknown camera motion $\mathbf{T}^* \in \mathrm{SE}(3)$ between two camera frames by minimizing the SSD as follows:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{\mathbf{u}\in\Omega_h} \Big(I_t(\mathbf{u}') - I_h(\mathbf{u})\Big)^2, \tag{5.1}$$

where $\Omega_h \subset \mathbb{R}^2$ is the set of all image coordinates in the host frame with valid inverse distance values, and $\mathbf{u}'$ the transformed pixel coordinates

$$\mathbf{u}' = w(\mathbf{T}, \mathbf{u}) = \pi_{\mathbf{c}}\Big(\widetilde{g}_d\big(\mathbf{T}, \pi_{\mathbf{c}}^{-1}(\mathbf{u}), d_{\mathbf{p}}\big)\Big), \tag{5.2}$$

$$= \pi_{\mathbf{c}}\big(\widetilde{g}_d(\mathbf{T}, \widetilde{\mathbf{p}}_d)\big), \tag{5.3}$$

$$= \pi_{\mathbf{c}}(\widetilde{\mathbf{p}}'_d). \tag{5.4}$$

In the above equations, $d_{\mathbf{p}}$ denotes the inverse distance associated to the pixel coordinates $\mathbf{u}$ in the host frame, whereas $\widetilde{\mathbf{p}}_d$ are the unnormalized homogeneous coordinates of the 3D point relative to its host frame, as defined in (3.42), i.e.:

$$\widetilde{\mathbf{p}}_d = \begin{pmatrix} \mathbf{b} \\ d_{\mathbf{p}} \end{pmatrix}. \tag{5.5}$$

Further, $\widetilde{\mathbf{p}}'_d$ represents the transformed 3D point in unnormalized homogeneous coordinates (cf. Equation (3.43)):

$$\widetilde{\mathbf{p}}'_d = \begin{pmatrix} \mathbf{p}'_s \\ d_{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} x'_s \\ y'_s \\ z'_s \\ d_{\mathbf{p}} \end{pmatrix}, \tag{5.6}$$

and $\mathbf{T} \in \mathrm{SE}(3)$ is the relative transformation matrix mapping 3D points from host to target frame, i.e., $\mathbf{T} = \mathbf{T}_{th}$. The rigid-body transformation of a 3D point in its (unnormalized) homogeneous coordinates $\widetilde{g}_d(\widetilde{\mathbf{p}}_d)$ was presented in (3.45), while the projection function $\pi_{\mathbf{c}}(\widetilde{\mathbf{p}}'_d)$ of a point in its homogeneous coordinates (either normalized or unnormalized) was defined in (3.48). Finally, w is the warping function that combines all transformations mapping a pixel $\mathbf{u}$ in the host frame to $\mathbf{u}'$ in the target frame.

DIA is a non-linear least squares problem, but a distinct minimum is often available as demonstrated in [30]. As presented in Section 3.5, to solve a non-linear least squares problem we can perform a linearization around a point using a Taylor expansion and solve the resulting normal equations. This is performed iteratively until convergence.

In Baker and Matthews [3] equivalent formulations for the linearization of the DIA problem are presented. Next, we describe two of them, namely the Forward Additive (FA) and the Inverse Compositional (IC) approaches.

5.1.1. Forward Additive Algorithm

In the FA formulation the problem is relinearized around the last estimate at every iteration. This approach assumes that a current estimate of the transformation parameter $\mathbf{T}$ is known and then iteratively solves for increments $\boldsymbol{\xi}$ to the parameters. In this way, the incrementally solved residual term amounts to

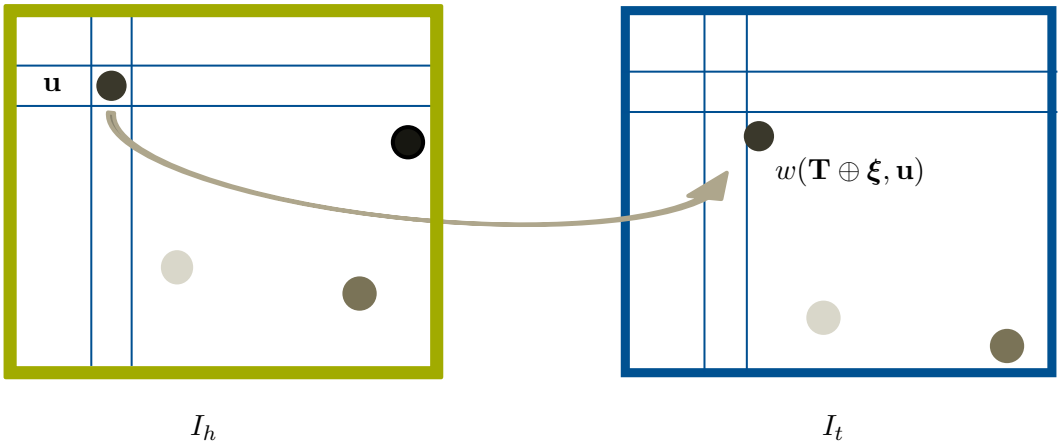


Figure 5.1.: Schematic representation of the FA approach, where I_h and I_t represent the host and target frames, respectively.

$$r_i(\mathbf{T} \oplus \boldsymbol{\xi}) = I_t(w(\mathbf{T} \oplus \boldsymbol{\xi}, \mathbf{u})) - I_h(\mathbf{u}), \tag{5.7}$$

where we use a left-multiplicative formulation to concatenate the increments $\widehat{\boldsymbol{\xi}} \in \mathrm{se}(3)$ (recall $\boldsymbol{\xi} \in \mathbb{R}^6$) with the previous estimate $\mathbf{T} \in \mathrm{SE}(3)$, i.e.,:

$$\mathbf{T} \oplus \boldsymbol{\xi} := \exp(\widehat{\boldsymbol{\xi}})\mathbf{T}. \tag{5.8}$$

Figure 5.1 gives a schematic view of the FA approach.

At every iteration we linearize the residuals around the last estimate $\mathbf{T}$ (cf. Equation (3.78)), which gives

$$r_{\mathrm{lin},i}(\mathbf{T} \oplus \boldsymbol{\xi}) = r_i(\mathbf{T}) + \mathbf{J}_i\boldsymbol{\xi}, \tag{5.9}$$

where

$$\begin{aligned} r_i(\mathbf{T}) &= I_t(\mathbf{u}') - I_h(\mathbf{u}) && (5.10)\\ &= I_t(w(\mathbf{T}, \mathbf{u})) - I_h(\mathbf{u}) && (5.11)\\ &= I_t\Big(\pi_{\mathbf{c}}\big(\underbrace{\widetilde{g}_d(\mathbf{T}, \pi_{\mathbf{c}}^{-1}(\mathbf{u}), d_{\mathbf{p}})}_{\widetilde{\mathbf{p}}'_d}\big)\Big) - I_h(\mathbf{u}). && (5.12) \end{aligned}$$

The solution to the least squares problem built upon the above linearized residual is

$$\boldsymbol{\xi} = -\sum_{\mathbf{u}\in\Omega_h} \mathbf{H}^{-1}\mathbf{J}_i^\top \left[I_t(\mathbf{u}') - I_h(\mathbf{u})\right], \tag{5.13}$$

where $\mathbf{H}$ is the Hessian matrix and has the form

$$\mathbf{H} = \sum_{\mathbf{u}\in\Omega_h} \mathbf{J}_i^\top \mathbf{W}\mathbf{J}_i. \tag{5.14}$$

Note that $\mathbf{J}_i$ is the 1×6 Jacobian of the i^{th} residual with respect to an increment $\boldsymbol{\xi}$ to the parameters $\mathbf{T}$ evaluated at $\boldsymbol{\xi}=\mathbf{0}$. Applying the chain rule we can decompose it into a product of Jacobians

$$\begin{aligned}\mathbf{J}_i &= \mathbf{J}_I\mathbf{J}_\pi\mathbf{J}_g && (5.15)\\ &= \frac{\partial I_t(\mathbf{u})}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}'=\pi_\mathbf{c}(\widetilde{\mathbf{p}}'_d)} \cdot \frac{\partial \pi_\mathbf{c}(\widetilde{\mathbf{p}})}{\partial \widetilde{\mathbf{p}}}\bigg|_{\widetilde{\mathbf{p}}=\widetilde{\mathbf{p}}'_d=\widetilde{g}_d(\mathbf{T},\widetilde{\mathbf{p}}_d)} \cdot \frac{\partial \widetilde{g}_d(\mathbf{T}\oplus\boldsymbol{\xi},\widetilde{\mathbf{p}}_d)}{\partial \boldsymbol{\xi}}\bigg|_{\boldsymbol{\xi}=\mathbf{0}}, && (5.16)\end{aligned}$$

where $\mathbf{J}_I$ is the 1×2 image gradient in the x and y direction; $\mathbf{J}_\pi$ is the 2×4 Jacobian matrix of the projection function with respect to the (unnormalized) homogeneous coordinates of the transformed 3D point $\widetilde{\mathbf{p}}'_d$; and $\mathbf{J}_g$ is the 4×6 Jacobian matrix of the rigid body motion with respect to its six parameters (cf. Equation (3.50)).

The form of $\mathbf{J}_\pi$ depends on the camera model employed (cf. Section 3.3), while $\mathbf{J}_I$ and $\mathbf{J}_g$ present the same form regardless of the camera model. Assuming, for instance, a pinhole camera model for $\mathbf{J}_\pi$, the three Jacobians are

$$\mathbf{J}_I = \frac{\partial I_t(\mathbf{u})}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}'=\pi_\mathbf{c}(\widetilde{\mathbf{p}}'_d)} = \begin{pmatrix} \nabla I_{t,x} & \nabla I_{t,y}\end{pmatrix}, \tag{5.17}$$

$$\mathbf{J}_\pi = \frac{\partial \pi_\mathbf{c}(\widetilde{\mathbf{p}})}{\partial \widetilde{\mathbf{p}}}\bigg|_{\widetilde{\mathbf{p}}=\widetilde{\mathbf{p}}'_d=\widetilde{g}_d(\mathbf{T},\widetilde{\mathbf{p}}_d)} = \begin{pmatrix} f_x\frac{1}{z'_s} & 0 & -f_x\frac{x'_s}{z'^2_s} & 0\\ 0 & f_y\frac{1}{z'_s} & -f_y\frac{y'_s}{z'^2_s} & 0\end{pmatrix}, \tag{5.18}$$

$$\mathbf{J}_g = \frac{\partial \widetilde{g}_d(\mathbf{T}\oplus\boldsymbol{\xi},\widetilde{\mathbf{p}}_d)}{\partial \boldsymbol{\xi}}\bigg|_{\boldsymbol{\xi}=\mathbf{0}} = \begin{pmatrix} d_\mathbf{p} & 0 & 0 & 0 & z'_s & -y'_s\\ 0 & d_\mathbf{p} & 0 & -z'_s & 0 & x'_s\\ 0 & 0 & d_\mathbf{p} & y'_s & -x'_s & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}, \tag{5.19}$$

where $\mathbf{J}_g$ can be derived as follows:

$$\mathbf{J}_g = \frac{\partial g(\mathbf{T}\oplus\boldsymbol{\xi},\widetilde{\mathbf{p}}_d)}{\partial \boldsymbol{\xi}}\bigg|_{\boldsymbol{\xi}=\mathbf{0}} = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\exp(\widehat{\boldsymbol{\xi}})\mathbf{T}\widetilde{\mathbf{p}}_d - \mathbf{T}\widetilde{\mathbf{p}}_d}{\boldsymbol{\xi}}. \tag{5.20}$$

By further performing a first order approximation of $\exp(\widehat{\boldsymbol{\xi}})$

$$\exp(\widehat{\boldsymbol{\xi}}) \approx \mathbf{I} + \widehat{\boldsymbol{\xi}} \tag{5.21}$$

we can write

$$\mathbf{J}_g = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{(\mathbf{I}+\widehat{\boldsymbol{\xi}})\mathbf{T}\widetilde{\mathbf{p}}_d - \mathbf{T}\widetilde{\mathbf{p}}_d}{\boldsymbol{\xi}} = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\widehat{\boldsymbol{\xi}}\,\mathbf{T}\widetilde{\mathbf{p}}_d}{\boldsymbol{\xi}} \overset{(3.43)}{=} \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\widehat{\boldsymbol{\xi}}\,\widetilde{\mathbf{p}}'_d}{\boldsymbol{\xi}}. \tag{5.22}$$

Plugging the definition of the twist (cf. Equation (3.53)) and the definition of $\widetilde{\mathbf{p}}'_d$ (cf. Equation (3.43)) into (5.22) gives

$$\mathbf{J}_g = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\begin{pmatrix} \widehat{\mathbf{w}} & \boldsymbol{\nu} \\ \mathbf{0} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{p}'_s \\ d_{\mathbf{p}} \end{pmatrix}}{\begin{pmatrix} \boldsymbol{\nu} \\ \mathbf{w} \end{pmatrix}} = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\begin{pmatrix} \widehat{\mathbf{w}}\mathbf{p}'_s + \boldsymbol{\nu} d_{\mathbf{p}} \\ 0 \end{pmatrix}}{\begin{pmatrix} \boldsymbol{\nu} \\ \mathbf{w}. \end{pmatrix}} = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\begin{pmatrix} -\widehat{\mathbf{p}'_s}\mathbf{w} + \boldsymbol{\nu} d_{\mathbf{p}} \\ 0 \end{pmatrix}}{\begin{pmatrix} \boldsymbol{\nu} \\ \mathbf{w} \end{pmatrix}}, \tag{5.23}$$

where in the last step we have inverted the order of the cross-product $\widehat{\mathbf{w}}\mathbf{p}'_s$, since by definition we have $\widehat{\mathbf{w}}\mathbf{p}'_s = -\widehat{\mathbf{p}'_s}\mathbf{w}$. Solving the limit gives us the final expression for $\mathbf{J}_g$:

$$\mathbf{J}_g = \begin{pmatrix} d_{\mathbf{p}}\mathbf{I} & -\widehat{\mathbf{p}'_s} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{5.24}$$

which corresponds to that presented already in Equation (5.19).

Note that all three Jacobians, namely $\mathbf{J}_I$, $\mathbf{J}_\pi$ and $\mathbf{J}_g$ have to be recomputed across iterations, since they all depend on the transformed point $\widetilde{\mathbf{p}}'_d$ or its projection $\mathbf{u}'$ onto the target frame.

5.1.2. Inverse Compositional

The IC approach also assumes that a current estimate of $\mathbf{T}$ is available. However, it iteratively solves for an incremental warp $w(\boldsymbol{\xi}, \mathbf{u})$ instead of an additive update $\boldsymbol{\xi}$ to $\mathbf{T}$ [2]. Moreover, the roles of the host and target frames are reversed. Therefore, instead of linearizing

$$r_i(\mathbf{T} \oplus \boldsymbol{\xi}) = I_t(w(\mathbf{T} \oplus \boldsymbol{\xi}, \mathbf{u})) - I_h(\mathbf{u}) \tag{5.25}$$

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

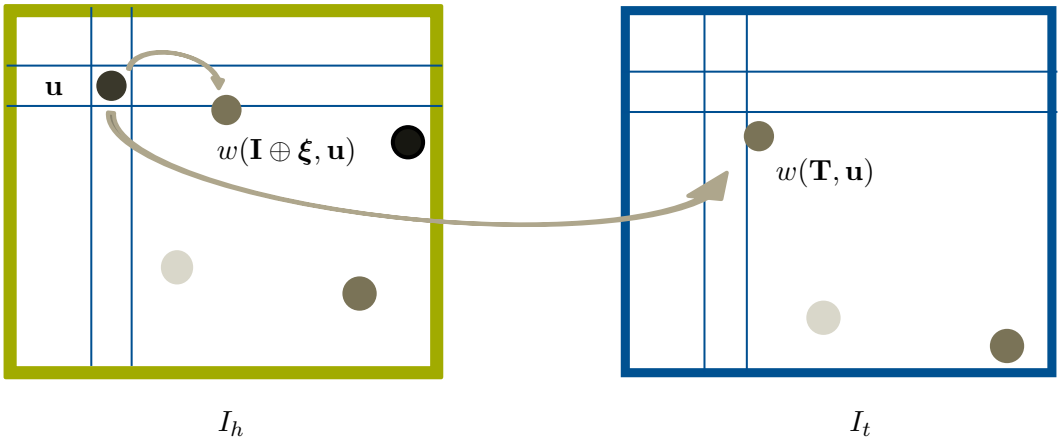


Figure 5.2.: Schematic representation of the IC approach, where I_h and I_t represent the host and target frames, respectively.

$$r_i(\boldsymbol{\xi}) = I_h(w(\mathbf{I} \oplus \boldsymbol{\xi}, \mathbf{u})) - I_t(w(\mathbf{T}, \mathbf{u})), \tag{5.26}$$

where $\mathbf{I}$ is the identity transform. Figure 5.2 provides a schematic view of the IC approach.

Performing a first order Taylor expansion on the above residual term gives

$$r_{\text{lin},i}(\boldsymbol{\xi}) = \underbrace{I_h(w(\mathbf{I} \oplus \mathbf{0}, \mathbf{u}))}_{I_h(\mathbf{u})} + \mathbf{J}_i\big|_{(\boldsymbol{\xi}=\mathbf{0},\mathbf{u})}\boldsymbol{\xi} - I_t(w(\mathbf{T}, \mathbf{u})), \tag{5.27}$$

where we have assumed without loss of generality that $w(\mathbf{I} \oplus \mathbf{0}, \mathbf{u})$ is the identity. The solution to the least squares problem built upon the linearized residual in (5.27) is

$$\boldsymbol{\xi} = -\sum_{\mathbf{u}\in\Omega_h} \mathbf{H}^{-1}\mathbf{J}_i^\top \left[I_h(\mathbf{u}) - I_t(w(\mathbf{T}, \mathbf{u}))\right] \tag{5.28}$$

$$= \sum_{\mathbf{u}\in\Omega_h} \mathbf{H}^{-1}\mathbf{J}_i^\top \left[I_t(w(\mathbf{T}, \mathbf{u})) - I_h(\mathbf{u})\right], \tag{5.29}$$

where $\mathbf{H}$ is the Hessian matrix, i.e.:

$$\mathbf{H} = \sum_{\mathbf{u}\in\Omega_h} \mathbf{J}_i^\top \mathbf{W} \mathbf{J}_i. \tag{5.30}$$

Note that $\mathbf{J}_i$ does not depend on the current parameter estimate $\mathbf{T}$. On the contrary, it is constant across iterations and can be precomputed once per pyramid level. Again, particularizing $\mathbf{J}_\pi$ for the case of a pinhole camera model, the resulting Jacobians in the IC approach are

$$\mathbf{J}_I = \left.\frac{\partial I_h(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}} = \begin{pmatrix} \nabla I_{h,x} & \nabla I_{h,y} \end{pmatrix}, \tag{5.31}$$

$$\mathbf{J}_\pi = \left.\frac{\partial \pi_{\mathbf{c}}(\widetilde{\mathbf{p}})}{\partial \widetilde{\mathbf{p}}}\right|_{\widetilde{\mathbf{p}}=\widetilde{\mathbf{p}}_d} = \begin{pmatrix} f_x \frac{1}{b_z} & 0 & -f_x \frac{b_x}{b_z^2} & 0 \\ 0 & f_y \frac{1}{b_z} & -f_y \frac{b_y}{b_z^2} & 0 \end{pmatrix}, \tag{5.32}$$

$$\mathbf{J}_g = \left.\frac{\partial \widetilde{g}_d(\boldsymbol{\xi}, \widetilde{\mathbf{p}}_d)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} = \begin{pmatrix} d_{\mathbf{p}} & 0 & 0 & 0 & b_z & -b_y \\ 0 & d_{\mathbf{p}} & 0 & -b_z & 0 & b_x \\ 0 & 0 & d_{\mathbf{p}} & b_y & -b_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{5.33}$$

Therefore, the IC algorithm amounts to iterating through the following four steps until convergence, as described in [2]:

1. Compute $I_t(w(\mathbf{T}, \mathbf{u}))$.
2. Compute $I_t(w(\mathbf{T}, \mathbf{u})) - I_h(\mathbf{u})$.
3. Re-evalutate the product $\mathbf{J}_i^\top \mathbf{W} \mathbf{J}_i$, since the weights $\mathbf{W}$ change across iterations.
4. Compute $\boldsymbol{\xi}$ using Equation (5.29).
5. Update the current warp: $w(\mathbf{T}, \mathbf{u}) = w(\mathbf{T}, \mathbf{u}) \circ w(\boldsymbol{\xi}, \mathbf{u})^{-1}$.

Updating the warp amounts in this particular case to left-incrementing the identity transform with the computed increment $\boldsymbol{\xi}$ and then right-multiplying the inverse of the resulting $\mathrm{SE}(3)$ element to the current estimate as follows:

$$\mathbf{T} = \mathbf{T}(\mathbf{I} \oplus \boldsymbol{\xi})^{-1}. \tag{5.34}$$

5.2. Keyframe Creation

Similarly to DSO and DSM, we combine different criteria to determine if a tracked frame needs to become a keyframe:

1. A new keyframe is required if the field of view changes, which we can detect by computing the mean square optical flow from the latest keyframe to the latest frame

$$f := \left(\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{u} - \mathbf{u}'\|^2 \right)^{\frac{1}{2}}. \tag{5.35}$$

2. Camera translation, while not necessarily changing the field of view, can cause occlusions and disocclusions. This can be measured by the mean optical flow without rotation

$$f_t := \left(\frac{1}{n} \sum_{i=1}^{n} \| \mathbf{u} - \mathbf{u}'_t \|^2 \right)^{\frac{1}{2}}, \tag{5.36}$$

   where $\mathbf{u}'_t$ is the transformed pixel position with $R = I_{3\times3}$, i.e., no rotation.

3. Another way to compute parallax between the tracked frame and the latest keyframe is through the product p between the estimated translation vector $\mathbf{t}$ and the mean inverse distance $\bar{\rho}$ of the keyframe's inverse distance map, i.e.:

$$p = \| \mathbf{t}\bar{\rho} \|. \tag{5.37}$$

These criteria are easy to obtain after initial frame alignment, and a weighted sum over these quantities allows us to determine if the tracked frame is required as keyframe. In particular, a new keyframe is spawned if

$$w_f \, f + w_{f_t} \, f_t + w_p \, p > 1. \tag{5.38}$$

After a new keyframe is created, new candidate points are selected in this newly added keyframe, which we cover in Section 6.1.1. Moreover, the LCW of active keyframes is recomputed, as it was described in Section 4.1.

5.3. Keyframe Removal

As described in Section 4.1, we aim to maintain a fixed number of temporal keyframes in the LCW. Therefore, after creating a new keyframe, recomputing the LCW, running PBA, removing outliers and projecting all active landmarks into the newly added keyframe (to create the tracking reference), we will select a keyframe from the temporal window for removal. We follow the strategy presented in [13] to select a keyframe for removal. Let $I_0 \dots I_n$ be the set of active keyframes, with I_0 being the newly added keyframe (see also stage 3 in Figure 4.2):

1. Apart from the newly added keyframe I_0, we always keep the latest two keyframes I_1 and I_2, which avoids premature fixation of keyframes' poses before they can be well optimized.

2. To have keyframes evenly distributed across space, we drop the keyframe I_i that maximizes

$$s(I_i) = \sqrt{d(I_1, I_i)} \sum_{j=3}^{N_t} (d(I_i, I_j) + \epsilon)^{-1}, \tag{5.39}$$

where $d(I_i, I_j)$ is the Euclidean distance between keyframes I_i and I_j, and ϵ a small constant. This strategy keeps active keyframes well-distributed in 3D space, rendering high parallax between them and therefore increasing accuracy.

As mentioned above, before removing the selected temporal keyframe, we project into the new keyframe (I_0) all landmarks hosted in the keyframes that were active in the last PBA, including that selected for removal. By doing so, we generate a new tracking reference for the alignment of subsequent frames.

6. Back-End

6.1. Point Management

As described in [13], image data is highly redundant, and in general simply using as much data points as possible does not necessarily bring about more benefits. On the contrary, Engel et al. show that sampling from from across all available data, including weakly textured or repetitive regions and edges, does increase performance.

In this work, we also aim at maintaining a fixed number N_l of active landmarks ($N_l = 2000$ in our experiments) that are distributed equally across space and active keyframes. In every new keyframe, we initially select N_{cps} candidate points (Section 6.1.1), which are then individually tracked in subsequent frames in order to obtain an initial coarse estimate of their inverse distance (Section 6.1.2). Finally, distinctive candidate points with low uncertainty in their inverse distance estimate might be activated (i.e., upgraded from *candidate point* to *landmark*) and therefore added to the optimization if required (Section 6.1.3).

6.1.1. Candidate Point Selection

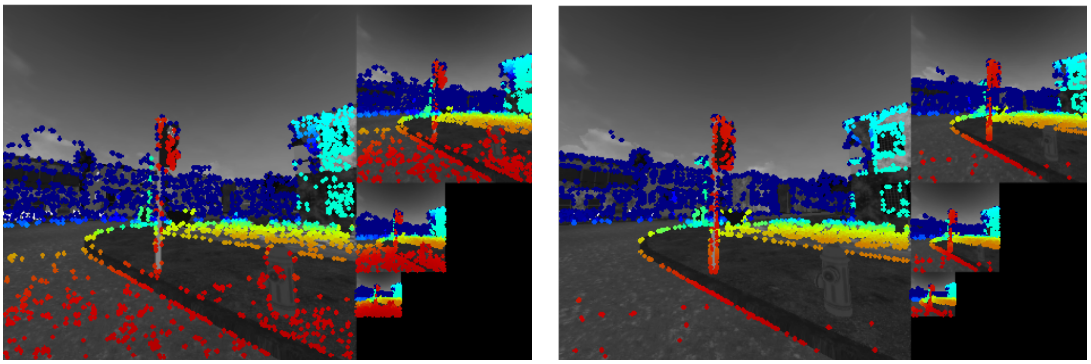
For every new keyframe, we aim at selecting N_{cps} candidate points (where $N_{cps} = 2000$ in our experiments) that are (1) well-distributed across the image and (2) locally salient in terms of their gradient magnitude.

To this end, we first split the image into a grid of $N \times N$-sized regions (we use $N = 64$ in our experiments). A gradient threshold g_i is computed for each region i individually as the sum of the median absolute gradient $\bar{g}_i$ of all pixels within the region and a global constant g_{th}, i.e.:

$$g_i = \bar{g}_i + g_{th}, \tag{6.1}$$

where $g_{th} = 7$ in our experiments (following DSO's recommendation).

We then divide the image into blocks of size $d \times d$ and select the image location within the current block that surpasses the gradient threshold g_i of the region in which the block resides. If the threshold is not surpassed, no candidate is selected for the current block.



(a) Setting A **(b)** Setting B

Figure 6.1.: Candidate point selection. Using a gradient-based threshold for a given region i of $\bar{g}_i + g_{th}$ allows us to select points that are both well-distributed across the image and with a sufficiently high gradient magnitude relative to the surrounding pixels (a). On the other hand, employing a threshold four times greater, and therefore being more strict, produces an unequal distribution of points (b).

As in DSO, the block-size d is recomputed continuously so as to produce the desired amount of candidate points. If too many points were created using a certain block-size for a given keyframe, two things will take place: on the one hand, for the current keyframe we will subsample the desired number of candidates from the initial (too large) set; on the other hand, we will reduce d such that for the next keyframe fewer candidates are selected already from the beginning (hopefully without the need to perform a subsequent subsampling).

Figure 6.1 shows two examples of candidate point selection, and how the threshold $\bar{g}_i$ described above plays an important role in selecing well-distributed points across the whole image.

6.1.2. Candidate Point Tracking

After tracking a new frame, for each active keyframe in the temporal window we track each of its candidate points individually by searching along the epipolar curve in the newly tracked frame and minimizing the photometric error (6.21). An uncertainty measure can additionally be computed, such that it constrains the discrete search for the candidate point in the subsequent tracked frame. This tracing procedure gives us an inverse distance for each candidate point, which will serve as a coarse initialization when upgrading the candidate to landmark and activating it for the PBA.

In [50] and later in [42] it was presented how to perform stereo matching for any central camera model, where epipolar lines are actually curves [23]. Similarly to [5, 42], we consider two points $\mathbf{p}^s_{min}, \mathbf{p}^s_{max} \in \mathbb{R}^3$ that lie on the unit sphere (see Figure 6.2) around the center of projection $\mathcal{C}_{cur}$ of the current tracked frame:

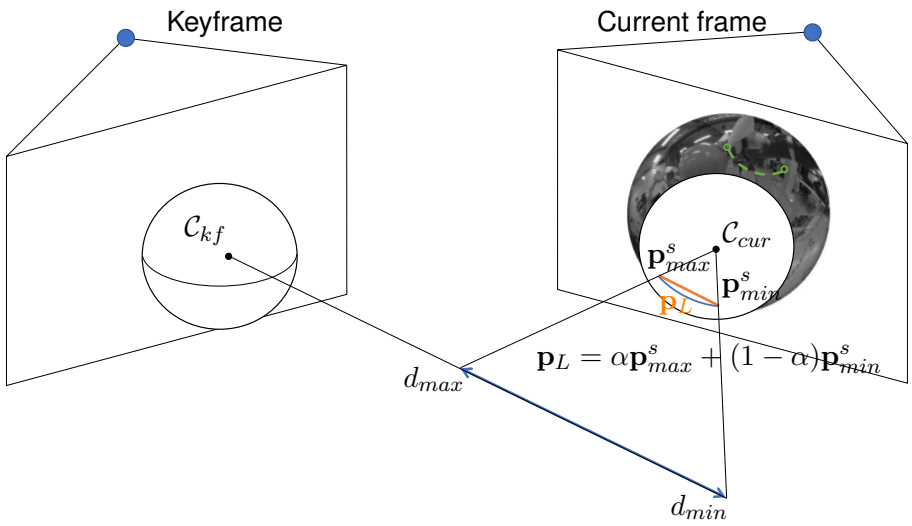


Figure 6.2.: Stereo matching for central camera models, based on that presented in [42].

$$\mathbf{p}^s_{min} \triangleq \pi_s(\mathbf{MT}\widetilde{\mathbf{p}}_{d_{min}}), \tag{6.2}$$
$$\mathbf{p}^s_{max} \triangleq \pi_s(\mathbf{MT}\widetilde{\mathbf{p}}_{d_{max}}), \tag{6.3}$$

which correspond to the minimum and maximum inverse distances d_{min}, d_{max} of the search interval. $\mathbf{T} \in \mathrm{SE}(3)$ is the relative transformation matrix mapping 3D points from the keyframe to the newly tracked frame, whereas $\mathbf{M}$ extracts the first three coordinates of a 3D point in its homogeneous coordinates (cf. Equation (3.46)). The map

$$\pi_s : \mathbb{R}^3 \to \mathbb{R}^3; \quad \mathbf{p} \mapsto \mathbf{p}^s = \frac{\mathbf{p}}{\|\mathbf{p}\|} \tag{6.4}$$

projects a 3D point onto the unit sphere, and $\widetilde{\mathbf{p}}_d = (b_x, b_y, b_z, d_\mathbf{p})^\top$ are the unnormalized homogeneous coordinates of the point relative to its host frame (cf. Equation (3.42)). The epipolar curve is then defined as the projection $\mathbf{u}_L(\alpha)$ of the linearly interpolated line $\mathbf{p}_L(\alpha)$ between $\mathbf{p}^s_{min}$ and $\mathbf{p}^s_{max}$ into the image plane

$$\mathbf{u}_L(\alpha) \triangleq \pi_\mathbf{c}(\mathbf{p}_L(\alpha)), \tag{6.5}$$

where

$$\mathbf{p}_L(\alpha) \triangleq \alpha\mathbf{p}^s_{max} + (1-\alpha)\mathbf{p}^s_{min} \quad \text{with} \quad \alpha \in [0,1]. \tag{6.6}$$

Under this framework, searching along the epipolar curve amounts to starting at $\mathbf{u}_L(0)$ and incrementing α every iteration in such a way that the corresponding pixel increment on the image plane equals approximately 1 pixel. To this end, we can employ the inverse of the norm of a first-order Taylor approximation of $\mathbf{u}_L$ to obtain an increment to α, which we denote by $\delta\alpha$, i.e:

$$\delta\alpha \approx \left\| \mathbf{J}_{\mathbf{u}_L}\big|_{\alpha} \right\|^{-1}, \tag{6.7}$$

with:

$$\mathbf{J}_{\mathbf{u}_L}\Big|_{\alpha} = \frac{\partial \mathbf{u}_L}{\partial \alpha}\Big|_{\alpha} = \frac{\partial \pi_{\mathbf{c}}}{\partial \mathbf{p}_L}\Big|_{\mathbf{p}_L(\alpha)} \cdot \frac{\partial \mathbf{p}_L(\alpha)}{\partial \alpha}\Big|_{\alpha} \tag{6.8}$$

$$= \frac{\partial \pi_{\mathbf{c}}}{\partial \mathbf{p}_L(\alpha)}\Big|_{\mathbf{p}_L} \cdot (\mathbf{p}^s_{max} - \mathbf{p}^s_{min}). \tag{6.9}$$

Note that $\frac{\partial \pi_{\mathbf{c}}}{\partial \mathbf{p}_L}\big|_{\mathbf{p}_L(\alpha)}$ depends on the camera model. The relation between all the above variables is visualized in Figure 6.2.

Inverse Distance Computation from the Best Match

Searching along the epipolar curve and looking for the image location that produces the lowest photometric error allows us to determine the best position on the image plane of the 3D point's observation in the new keyframe. The next step is to compute the inverse distance of the underlying 3D point $\mathbf{p}$ with respect to its host frame, i.e., $d_{\mathbf{p}}$.

From Section 3.4.1, and assuming for now $d_{\mathbf{p}} \neq 0$, we know that the transformed point $\mathbf{p}'$ relative to the current tracked frame can be computed as follows:

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{6.10}$$

$$= \mathbf{R}\frac{\mathbf{b}}{d_{\mathbf{p}}} + \mathbf{t}, \tag{6.11}$$

where $\mathbf{b} = \pi_{\mathbf{c}}^{-1}(\mathbf{u})$ (cf. Equation (3.6)) is the bearing vector of point $\mathbf{p}$. Moreover, we can compute the transformed bearing vector $\mathbf{b}'$ (again, relative to the current tracked frame) as

$$\mathbf{b}' = \frac{\mathbf{p}'}{\|\mathbf{p}'\|} = \frac{\mathbf{R}\frac{\mathbf{b}}{d_{\mathbf{p}}} + \mathbf{t}}{\left\|\mathbf{R}\frac{\mathbf{b}}{d_{\mathbf{p}}} + \mathbf{t}\right\|}. \tag{6.12}$$

Figure 6.3 shows a schematic view of the geometric relationship between the above variables. For conciseness, let $\mathbf{r}_0$, $\mathbf{r}_1$ and $\mathbf{r}_2$ represent the rows of the rotation matrix $\mathbf{R}$, and t_x, t_y and t_z

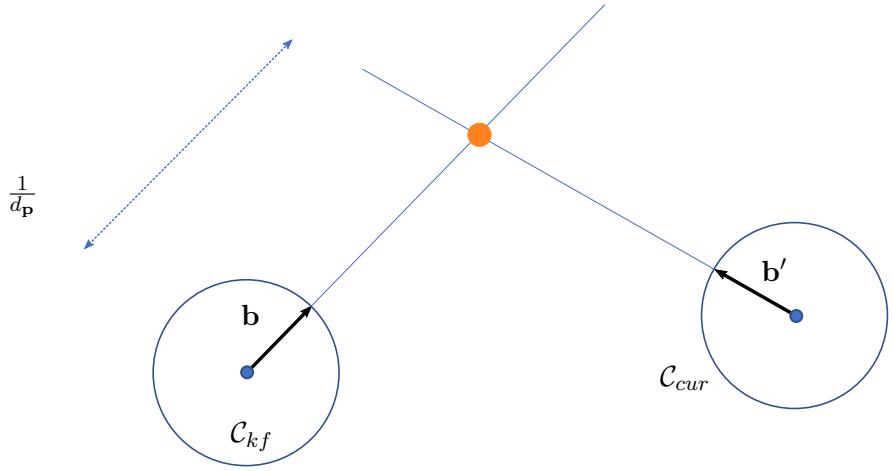


Figure 6.3.: Inverse distance computation from the best match after epipolar curve search.

the three components of the translation vector $\mathbf{t}$. Let us also compute, for instance, the first two components of $\mathbf{b}'$, namely

$$b'_x = \frac{p'_x}{\|\mathbf{p}'\|} = \frac{\frac{\mathbf{r}_0 \cdot \mathbf{b}}{d_{\mathbf{p}}} + t_x}{\|\mathbf{p}'\|} \tag{6.13}$$

and

$$b'_y = \frac{p'_y}{\|\mathbf{p}'\|} = \frac{\frac{\mathbf{r}_1 \cdot \mathbf{b}}{d_{\mathbf{p}}} + t_y}{\|\mathbf{p}'\|}. \tag{6.14}$$

Assuming for now that $b'_y \neq 0$, we can divide the former by the latter, obtaining

$$\frac{b'_x}{b'_y} = \frac{\frac{\mathbf{r}_0 \cdot \mathbf{b}}{d_{\mathbf{p}}} + t_x}{\frac{\mathbf{r}_1 \cdot \mathbf{b}}{d_{\mathbf{p}}} + t_y}. \tag{6.15}$$

Rewriting the above equation as

$$b'_x \frac{\mathbf{r}_1 \cdot \mathbf{b}}{d_\mathbf{p}} + b'_x t_y = b'_y \frac{\mathbf{r}_0 \cdot \mathbf{b}}{d_\mathbf{p}} + b'_y t_x \tag{6.16}$$

and multiply (6.16) by $d_\mathbf{p}$ gives

$$b'_x \mathbf{r}_1 \cdot \mathbf{b} + d_\mathbf{p} b'_x t_y = b'_y \mathbf{r}_0 \cdot \mathbf{b} + d_\mathbf{p} b'_y t_x. \tag{6.17}$$

We can finally solve for $d_\mathbf{p}$ (i.e., the inverse distance of the candidate point relative to its host keyframe) as follows:

$$d_\mathbf{p} = \frac{b'_y \mathbf{r}_0 \cdot \mathbf{b} - b'_x \mathbf{r}_1 \cdot \mathbf{b}}{b'_x t_y - b'_y t_x}. \tag{6.18}$$

Note that Equation (6.18) also works for infinitely far away points, for which $d_\mathbf{p} = 0$. In such case, we have that $\mathbf{b}' = \mathbf{R}\mathbf{b}$, which can be derived by making $d_\mathbf{p}$ tend to zero in Equation (6.12), i.e.:

$$\mathbf{b}'_{d_\mathbf{p} \to 0} = \lim_{d_\mathbf{p} \to 0} \frac{\mathbf{R}\frac{\mathbf{b}}{d_\mathbf{p}} + \mathbf{t}}{\left\| \mathbf{R}\frac{\mathbf{b}}{d_\mathbf{p}} + \mathbf{t} \right\|}. \tag{6.19}$$

We can multiply both the numerator and the denominator by $d_\mathbf{p}$, thus obtaining

$$\mathbf{b}'_{d_\mathbf{p} \to 0} = \lim_{d_\mathbf{p} \to 0} \frac{\mathbf{R}\mathbf{b} + d_\mathbf{p}\mathbf{t}}{\|\mathbf{R}\mathbf{b} + d_\mathbf{p}\mathbf{t}\|} = \lim_{d_\mathbf{p} \to 0} \frac{\mathbf{R}\mathbf{b}}{\|\mathbf{R}\mathbf{b}\|} = \mathbf{R}\mathbf{b}, \tag{6.20}$$

where $\|\mathbf{R}\mathbf{b}\| = \|\mathbf{R}\| \cdot \|\mathbf{b}\| = 1 \cdot 1 = 1$, which derives from the corresponding definitions of $\mathbf{R}$ and $\mathbf{b}$.

When $\mathbf{b}' = \mathbf{R}\mathbf{b}$, the numerator in Equation (6.18) becomes zero, and so does the resulting inverse distance $d_\mathbf{p}$, thus demonstrating the validity of (6.18) for infinitely far away points.

Note, however, that Equation (6.18) is not valid when b'_y is zero (cf. Equation (6.16)), nor when the denominator in (6.18) equates zero. The latter case can occur in either of the following cases:

1. if $b'_x = b'_y = 0$,
2. if $t_x = t_y = 0$,
3. or if $\mathbf{t}$ and $\mathbf{b}'$ are parallel in the x and y dimensions.

To ensure that none of the above cases take place we can exploit the fact that Equation (6.18) can be derived from any pair out of the triplet (b_x, b_y, b_z). Therefore, as denominator in Equation

(6.16) we will always choose the largest component of $\mathbf{b}'$.

Moreover, out of the remaining two components of $\mathbf{b}'$ we will select that which maximizes the value of the denominator in Equation (6.18), discarding the candidate point if in the best-case scenario the denominator is still too close to zero.

Note as well that non-zero translation in at least one dimension is also a straight forward assumption, since the epipolar curve search does not make sense otherwise.

Finally, if $\mathbf{b}'$ and $\mathbf{t}$ point in the exact same direction we cannot retrieve an inverse distance, which relates to the fact that estimating depth close to the focus of expansion is ill-posed.

6.1.3. Candidate Point Activation

When the LCW is recomputed after the creation of a new keyframe, candidate points hosted in any of the temporal keyframes might be upgraded to landmarks (and possibly also activated to be part of the next local PBA) if required.

Candidate points can have either of the following status:

- *Uninitialized*: the candidate point was not even traced once.
- *Skipped*: the point has already been traced successfully, to the extent that it was not traced in the last frame to avoid unnecessary computation.
- *BadConditioned*: the point was traced in previous frames, but not in the last one due to bad conditioning of the epipolar curve search for this point in particular.
- *OutOfBounds*: the point fell out of bounds in the last frame. It could still have been traced successfully in previous frames, and it may still be traced in the subsequent frame.
- *Outlier*: the point's photometric error in the last frame surpassed a threshold, but it is allowed to be traced in subsequent frames.
- *OutlierForSure*: the point's photometric error has surpassed a threshold in two consecutive frames, leading to the marking of the point as a fixed outlier.
- *Good*: the point was traced successfully in the last frame.

When activating candidate points for their addition to the PBA we go over all the temporal keyframes, starting from the newest (I_1 in Figure 4.2). At this moment, all candidate points that are marked as Outlier or OutlierForSure will be removed forever from the list of candidate points of the current temporal keyframe.

On the other hand, the candidate point is proposed for activation if it meets all of the following conditions:

1. The candidate point's status is either Skipped, BadConditioned, OutOfBounds or Good,

2. the quality of the discrete search is bigger than 3 (where we define quality as the photometric error ratio between the best and the second-best match),

3. the length in pixels of (a linear approximation of) the epipolar curve is smaller than 8 pixels,

4. and, finally, the candidate's inverse distance is non-negative.

When a candidate point meets all the above requirements, its inverse distance is then optimized against all active keyframes in the LCW in a structure-only PBA, i.e., camera poses are fixed. Only those candidates that can be optimized with low uncertainty against all active keyframes are further proposed for activation, where we use the inverse of the candidate point's 1×1 Hessian as an approximation of the point's covariance.

Furthermore, after this structure-only optimization we compute the photometric error of the candidate point in each of the active keyframes using the final estimate of $d_{\mathbf{p}}$, rejecting those observations which surpass a given threshold. This allows us not only to obtain the visibility of the candidate point in the LCW but also to prevent the introduction of high-residual observations into the full PBA, which otherwise would degrade the solution.

Finally, we will only upgrade the candidate point to the being a landmark if its projection into the new keyframe (I_0) falls into a depleted (unexplored) region of a previously built distance map of the new keyframe, where every pixel in such distance map indicates the distance in pixels to the closest projection of an active landmark.

Figure 6.4 shows an example sequence of candidate point tracking along three subsequently tracked frames. In this example, the third tracked frame is marked as keyframe. Consequently, the LCW is recomputed and new candidate points are upgraded to become landmarks (purple points on the last row of Figure 6.4), which will densify the previous map (blue points). Moreover, some of the candidate points that are not upgraded to landmark will be discarded and removed forever, but other (those potentially upgradable) will remain as candidate points (dark red points on the third row of Figure 6.4).

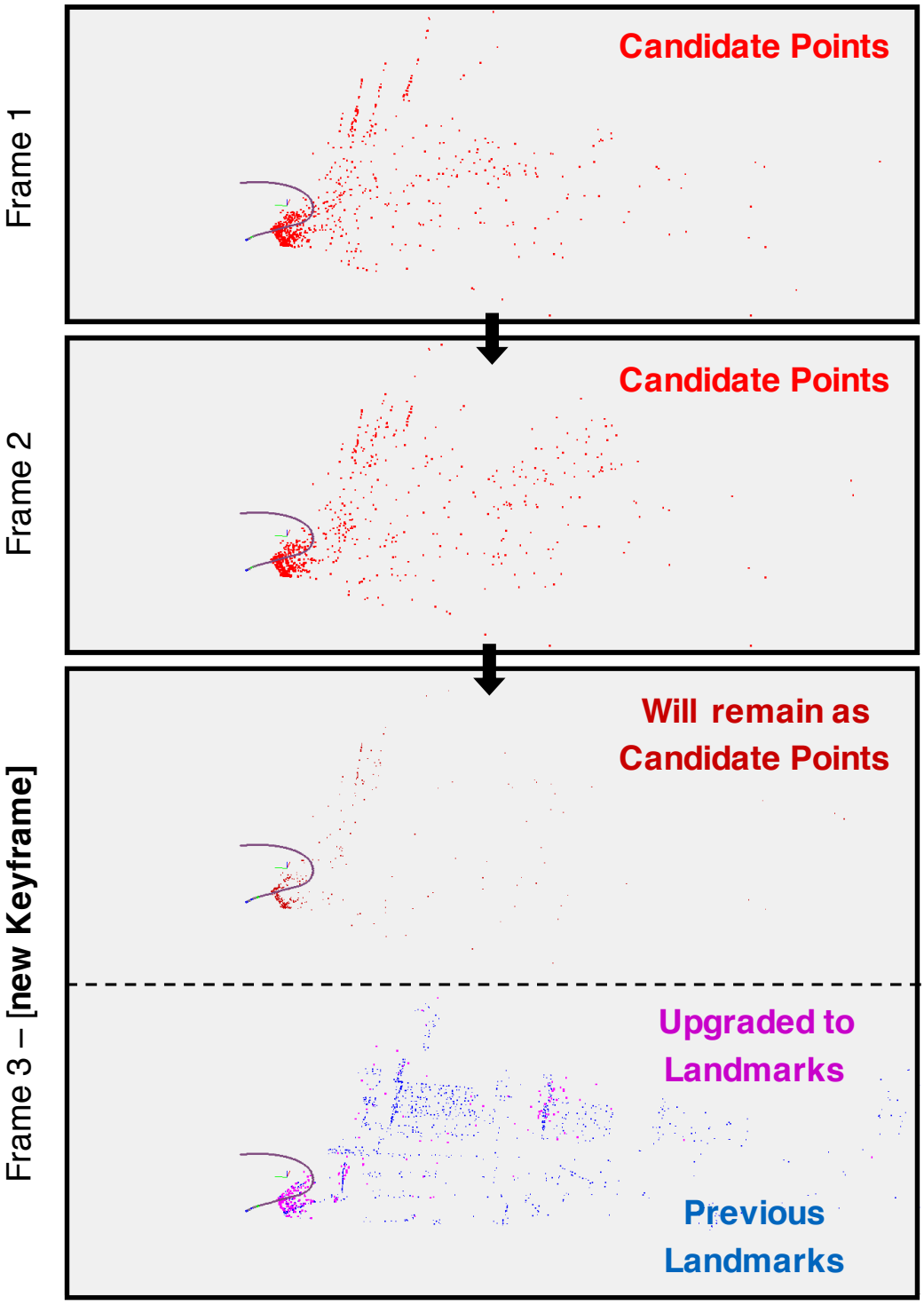


Figure 6.4.: Candidate point tracking and their upgrading to landmark.

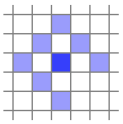


Figure 6.5.: Residual pattern. Pattern N_p proposed in [13] for energy computation, where the bottom-right pixel is omitted so that the number of pixels in the patch is a power of two for more efficient computation.

6.2. Photometric Bundle Adjustment

Following [13] we define the photometric error of a point $\mathbf{u}$ hosted in the keyframe I_h and observed in a target keyframe I_t as the weighted SSD over a small patch of pixels centered around pixel $\mathbf{u}$:

$$E_{\mathbf{u}} \triangleq \sum_{\mathbf{u}_i \in \mathcal{N}_{\mathbf{u}}} \rho\left(I_t(\mathbf{u}'_i) - I_h(\mathbf{u}_i)\right) = \sum_{\mathbf{u}_i \in \mathcal{N}_{\mathbf{u}}} \rho(r_i), \tag{6.21}$$

where $\mathcal{N}_{\mathbf{u}}$ is the set of pixels in the patch; $\rho(\cdot)$ is a robust error function (cf. Section 3.5.3) and $\mathbf{u}'_i$ is the transformed pixel position given by Equation (5.4). As in [13], we use 8 pixels arranged in a spread pattern as shown in Figure 6.5. According to [13], such a pattern provides a good trade-off between computational costs, robustness to motion blur and information gain.

In terms of notation, we will use $\boldsymbol{\zeta} \in \mathrm{SE}(3)^n \times \mathbb{R}^m$ to represent all optimized variables, i.e., n active keyframes and m active landmarks, giving a total of $d = 6n + m$ parameters. Furthermore, we will use $\boldsymbol{\epsilon} \in \mathrm{se}(3)^n \times \mathbb{R}^m$ to denote the increments to the parameters $\boldsymbol{\zeta}$. Incrementing the current state estimate is given by

$$\boldsymbol{\zeta}^{(t+1)} = \boldsymbol{\zeta}^{(t)} \oplus \boldsymbol{\epsilon}, \tag{6.22}$$

where the $\oplus$-operator presented in (5.8) has been extended here to all optimized parameters; for parameters other than $\mathrm{SE}(3)$ camera poses it represents conventional addition.

After the creation of a new keyframe, as already presented throughout previous sections, the LCW (cf. Figure 4.2) is recomputed and all model parameters are optimized by minimizing the the full photometric error over all active keyframes and active map points in the LCW, which we can write as

$$E_{photo} = \sum_{I_h \in \mathcal{K}} \sum_{\mathbf{u} \in U_h} \sum_{t \in obs(\mathbf{u})} E_{\mathbf{u}}, \tag{6.23}$$

where $\mathcal{K}$ is the set of all active keyframes in the LCW, U_h the set of all active landmarks in keyframe I_h and $obs(\mathbf{u})$ the set of all keyframes, either active temporal, active covisible or inactive

covisible (cf. Figure 4.2) that observe point $\mathbf{u}$.

Equation (6.23) can be minimized using the iteratively reweighted G-N algorithm, which provides a good trade-off between flexibility and speed, or the L-M algorithm, which interpolates between G-N and the method of gradient descent. In either case, to account for those residuals for which the initial solution is not within the convergence radius we employ a coarse-to-fine scheme, where the solution in each level is employed as initialization in the next (finer) level.

For a given level, starting from an initial estimate $\boldsymbol{\zeta}^{(0)}$, in each iteration t we compute the weights w_i (according to the robust error function ρ) and photometric residuals r_i to then estimate an increment $\boldsymbol{\epsilon}$ given by

$$\boldsymbol{\epsilon} = -\mathbf{H}^{-1}\mathbf{b}, \tag{6.24}$$

$$\mathbf{H} = \mathbf{J}^{\top}\mathbf{W}\mathbf{J}, \tag{6.25}$$

$$\mathbf{b} = \mathbf{J}^{\top}\mathbf{W}\mathbf{r}, \tag{6.26}$$

which results from solving for the minimum of a second order approximation of (6.23) with fixed weights (cf. Equation (3.96)). (For simplicity, we describe the standard G-N algorithm, but extending this derivation to the L-M algorithm is straightforward.)

In the above equations, $\mathbf{r} \in \mathbb{R}^p$ is the vector of all p residuals, $\mathbf{W} \in \mathbb{R}^{p \times p}$ a diagonal matrix with weights w_i and $\mathbf{J} \in \mathbb{R}^{p \times d}$ the Jacobian of the residuals $\mathbf{r}$ with respect to a left-composed increment to all model parameters d, where the Jacobian for the residual r_i (i.e., the i-th row of $\mathbf{J}$) is defined as follows:

$$\mathbf{J}_i = \left. \frac{\partial r_i(\boldsymbol{\zeta}^{(t)} \oplus \boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} \right|_{\boldsymbol{\epsilon}=\mathbf{0}}. \tag{6.27}$$

We have implemented a custom solver for the PBA, which allows to select between the G-N and the L-M algorithms. For comparison, we have also implemented PBA using the optimization framework Ceres [1]. Image gradients are computed using central pixel differences at integer positions, while bilinear interpolation is employed for subpixel intensity and gradient evaluation.

6.2.1. Relative-Absolute Pose Formulation

We use the relative formulation presented in [62] for our custom PBA solver, which allows us to perform most computations for a given host-target pair in parallel. In this formulation, we apply the Schur complement on relative poses first and then combine the solution to solve for the increments to the absolute poses. (Note that we do not need to differentiate between a relative and an absolute formulation for landmarks.)

Let us first denote the stacked vector of residuals $\mathbf{r}(\boldsymbol{\epsilon})$ as

$$\mathbf{r}(\boldsymbol{\epsilon}) = \begin{pmatrix} \vdots \\ \mathbf{r}^i(\mathbf{f}_a^i(\boldsymbol{\epsilon})) \\ \vdots \end{pmatrix}, \tag{6.28}$$

where each entry $\mathbf{r}^i(\mathbf{f}_a^i(\boldsymbol{\epsilon}))$ is in turn the vector of all residuals generated from observations of landmarks hosted in keyframe i. In the remainder of this section, all variables with the superscript i will be referring to host keyframe i. Moreover, $\mathbf{f}_a(\cdot)$ is a vector-valued function that selects all target frames for host frame i and expresses their poses relative to the host frame. Furthermore, it selects the landmarks hosted in frame i out of all landmarks in the problem.

When performing the linearization of the residuals we can decompose the full Jacobian $\mathbf{J}_\mathbf{r}$ (i.e., the Jacobian of the residuals $\mathbf{r}$ with respect to the parameter increments $\boldsymbol{\epsilon}$) into the product of the relative-pose Jacobians $\mathbf{J}_r$ (i.e., the Jacobians of the residuals $\mathbf{r}$ with respect to the relative poses and landmarks) and the absolute-pose Jacobians $\mathbf{J}_a$ (i.e., the Jacobians of $\mathbf{f}_a$ with respect to the parameter increments $\boldsymbol{\epsilon}$), thus giving for host keyframe i the following linearized residual:

$$\mathbf{r}_{\text{lin}}^i(\boldsymbol{\epsilon}) = \mathbf{r}_0^i + \mathbf{J}_\mathbf{r}^i \boldsymbol{\epsilon} = \mathbf{r}_0^i + \mathbf{J}_r^i \underbrace{\mathbf{J}_a^i \boldsymbol{\epsilon}}_{\boldsymbol{\epsilon}^i}, \tag{6.29}$$

where $\boldsymbol{\epsilon}^i$ encodes the relative pose increment for the target frames of (host) keyframe i, as well as the inverse distance increments for the active landmarks hosted in i, and $\mathbf{r}_0^i$ is the vector of residuals (for host keyframe i) at the evaluation point.

Let us now denote by n' the number of relative poses in which keyframe i is the host frame, and by m' the number of all active landmarks hosted in keyframe i. (Recall that we defined n as the total number of absolute poses in the PBA, and m as the total number of active landmarks.) We can therefore define the absolute-pose Jacobian $\mathbf{J}_a^i$ of host keyframe i as

$$\mathbf{J}_a^i = \frac{\partial \mathbf{f}_a^i(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} = \left(\begin{array}{c|c} \mathbf{J}_{pa}^i & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \cdots \mathbf{I} \cdots \mathbf{0} \end{array}\right) \in \mathbb{R}^{(6n'+m') \times (6n+m)}, \tag{6.30}$$

$$\mathbf{J}_{pa}^i = \begin{pmatrix} \vdots \\ — \quad \mathbf{J}_{pa}^{ij} \quad — \\ \vdots \end{pmatrix} \in \mathbb{R}^{6n' \times 6n}, \tag{6.31}$$

$$\mathbf{J}_{pa}^{ij} = \begin{pmatrix} \cdots & \frac{\partial \mathbf{T}_{th}^{ij}}{\partial \mathbf{T}_{wh}^{i}} & \cdots & \frac{\partial \mathbf{T}_{th}^{ij}}{\partial \mathbf{T}_{wt}^{j}} & \cdots \end{pmatrix} \in \mathbb{R}^{6 \times 6n}, \tag{6.32}$$

where $\mathbf{J}_{pa}^{ij}$ has 2 non-zero entries, one for the host keyframe i and other for the target keyframe j. Under this framework, we can now further decompose Equation (6.24) into

$$\underbrace{\sum_i \mathbf{H}^i}_{\mathbf{H}}(-\boldsymbol{\epsilon}) = \underbrace{\sum_i \mathbf{b}^i}_{\mathbf{b}}, \tag{6.33}$$

where $\mathbf{H} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\mathbf{b} \in \mathbb{R}^{n+m}$ result from summing over all host keyframes i. In particular, for a single host i we have

$$\mathbf{H}^i = \mathbf{J}_a^{i\top} \underbrace{\mathbf{J}_r^{i\top} \mathbf{W}^i \mathbf{J}_r^i}_{\mathbf{H}_r^i} \mathbf{J}_a^i, \tag{6.34}$$

$$\mathbf{b}^i = \mathbf{J}_a^{i\top} (\underbrace{\mathbf{J}_r^{i\top} \mathbf{W} \mathbf{r}_0^i}_{\mathbf{b}_r^i}). \tag{6.35}$$

In the above two equations we have implicitly defined $\mathbf{H}_r^i \in \mathbb{R}^{(6n'+m')\times(6n'+m')}$ and $\mathbf{b}_r^i \in \mathbb{R}^{6n'+m'}$, where the subscript r stands for relative (in turn, a stands for absolute). Solving the normal equation (6.33) can be simplified by first tackling the problem in its relative formulation, i.e.:

$$\mathbf{H}_r^i(-\boldsymbol{\epsilon}_r^i) = \mathbf{b}_r^i, \tag{6.36}$$

$$\underbrace{\begin{pmatrix} \mathbf{H}_{pp}^i & \mathbf{H}_{pl}^i \\ \mathbf{H}_{pl}^{i\top} & \mathbf{H}_{ll}^i \end{pmatrix}}_{\mathbf{H}_r^i} \underbrace{\begin{pmatrix} \mathbf{x}_r^i \\ \mathbf{y}^i \end{pmatrix}}_{-\boldsymbol{\epsilon}_r^i} = \underbrace{\begin{pmatrix} \mathbf{b}_p^i \\ \mathbf{b}_l^i \end{pmatrix}}_{\mathbf{b}_r^i}, \tag{6.37}$$

where $\mathbf{x}_r^i \in \mathbb{R}^{6n'}$ is comprised of $\mathrm{se}(3)$ parameter vectors, each representing a pose increment to a *relative* pose between host i and a target j. On the other hand, $\mathbf{y}^i \in \mathbb{R}^{m'}$ denotes the increments to be applied to the inverse distance of each active landmark in the host frame i. (Note that $\mathbf{y}^i$ has no subscript, since the inverse distance of a landmark does not have an absolute or relative formulation.) $\mathbf{H}_{pp}^i$ is a block diagonal matrix with 6×6 non-zero elements $\mathbf{H}_{pp}^{ij}$ in the diagonal, each corresponding to the relative pose between the host keyframe i and a target keyframe j:

$$\mathbf{H}_{pp}^i = \begin{pmatrix} \ddots & & \\ & \mathbf{H}_{pp}^{ij} & \\ & & \ddots \end{pmatrix} \in \mathbb{R}^{6n'\times 6n'}. \tag{6.38}$$

Matrix $\mathbf{H}_{ll}$ is also a diagonal matrix, where each entry $H_{ll}^{ik} \in \mathbb{R}$ corresponds to a landmark k hosted in keyframe i:

$$\mathbf{H}_{ll}^{i} = \begin{pmatrix} \ddots & & \\ & H_{ll}^{ik} & \\ & & \ddots \end{pmatrix} \in \mathbb{R}^{m' \times m'}. \tag{6.39}$$

Moreover, $\mathbf{H}_{pl}^{i} \in \mathbb{R}^{6n' \times m'}$ contains the hessian blocks relating all relative poses in which i is the host frame with every landmark k that i hosts:

$$\mathbf{H}_{pl}^{i} = \begin{pmatrix} & \vdots & \\ \cdots & \mathbf{H}_{pl}^{ij} & \cdots \\ & \vdots & \end{pmatrix}. \tag{6.40}$$

In turn, $\mathbf{H}_{pl}^{ij} \in \mathbb{R}^{6 \times m'}$ contains all pose-to-landmark hessian blocks for host i and target j:

$$\mathbf{H}_{pl}^{ij} = \begin{pmatrix} \cdots & \mathbf{H}_{pl}^{ijk} & \cdots \end{pmatrix}, \tag{6.41}$$

where $\mathbf{H}_{pl}^{ijk} \in \mathbb{R}^{6}$ is the hessian block corresponding to the landmark k hosted in i and being observed by the target frame j. Note that if landmark k is not observed by target frame j, then $\mathbf{H}_{pl}^{ijk} = 0$.

Onto this relative formulation presented above we can now apply the Schur complement [59], yielding the following for host keyframe i:

$$\mathbf{y}^{i} = {\mathbf{H}_{ll}^{i}}^{-1}(\mathbf{b}_{l}^{i} - {\mathbf{H}_{pl}^{i}}^{\top}\mathbf{x}_{r}^{i}), \tag{6.42}$$

$$\underbrace{(\mathbf{H}_{pp}^{i} - \mathbf{H}_{pl}^{i}{\mathbf{H}_{ll}^{i}}^{-1}{\mathbf{H}_{pl}^{i}}^{\top})}_{\mathbf{H}_{r}^{i*}}\mathbf{x}_{r}^{i} = \underbrace{\mathbf{b}_{p}^{i} - \mathbf{H}_{pl}^{i}{\mathbf{H}_{ll}^{i}}^{-1}\mathbf{b}_{l}^{i}}_{\mathbf{b}_{r}^{i*}}. \tag{6.43}$$

We can now go back to an absolute-pose formulation by means of applying Equations (6.34) and (6.35) onto (6.43) (i.e., pre- and post-multiplying $\mathbf{H}_{r}^{i*}$ by ${\mathbf{J}_{pa}^{i}}^{\top}$ and $\mathbf{J}_{pa}^{i}$, respectively, and pre-multiplying $\mathbf{b}_{r}^{i*}$ by ${\mathbf{J}_{pa}^{i}}^{\top}$) and summing over all host frames i in order to add up the effect that each relative pose has on its corresponding two absolute poses:

$$\underbrace{\left(\sum_{i} {\mathbf{J}_{pa}^{i}}^{\top}\mathbf{H}_{r}^{i*}\mathbf{J}_{pa}^{i}\right)}_{\mathbf{H}^{*}}\mathbf{x} = \underbrace{\sum_{i} {\mathbf{J}_{pa}^{i}}^{\top}\mathbf{b}_{r}^{i*}}_{\mathbf{b}^{*}}. \tag{6.44}$$

By solving normal equation (6.44) we obtain an absolute pose increment $-\mathbf{x}$ to each absolute pose in the PBA. Note that we could have arrived at the same result by having started from the

normal equation of the full system (which can be derived by plugging Equations (6.34) and (6.35) into (6.33)):

$$\underbrace{\left(\begin{array}{c|ccc} \sum_i \mathbf{J}_{pa}^{i\top}\mathbf{H}_{pp}^{i}\mathbf{J}_{pa}^{i} & \cdots & \mathbf{J}_{pa}^{i\top}\mathbf{H}_{pl}^{i} & \cdots \\ \hline \vdots & \ddots & & \\ \mathbf{H}_{pl}^{i\top}\mathbf{J}_{pa}^{i} & & \mathbf{H}_{ll}^{i} & \\ \vdots & & & \ddots \end{array}\right)}_{\mathbf{H}} \underbrace{\left(\begin{array}{c} \mathbf{x} \\ \hline \vdots \\ \mathbf{y}^{i} \\ \vdots \end{array}\right)}_{-\boldsymbol{\epsilon}} = \underbrace{\left(\begin{array}{c} \sum_i \mathbf{J}_{pa}^{i\top}\mathbf{b}_{p}^{i} \\ \hline \vdots \\ \mathbf{b}_{l}^{i} \\ \vdots \end{array}\right)}_{\mathbf{b}}. \tag{6.45}$$

Applying the Schur complement onto (6.45) yields

$$\mathbf{y}^{i} = \mathbf{H}_{ll}^{i\,-1}(\mathbf{b}_{l}^{i} - \mathbf{H}_{pl}^{i\top}\underbrace{\mathbf{J}_{pa}^{i}\mathbf{x}}_{\mathbf{x}^{i}}), \tag{6.46}$$

$$\left(\sum_i \mathbf{J}_{pa}^{i\top}\mathbf{H}_{pp}^{i}\mathbf{J}_{pa}^{i} - \sum_i \mathbf{J}_{pa}^{i\top}\mathbf{H}_{pl}^{i}\mathbf{H}_{ll}^{i\,-1}\mathbf{H}_{pl}^{i\top}\mathbf{J}_{pa}^{i}\right)\mathbf{x} = \sum_i \mathbf{J}_{pa}^{i\top}\mathbf{b}_{p}^{i} - \sum_i \mathbf{J}_{pa}^{i\top}\mathbf{H}_{pl}^{i}\mathbf{H}_{ll}^{i\,-1}\mathbf{b}_{l}^{i}, \tag{6.47}$$

where we can reorganize Equation (6.47) and then rewrite it in terms of $\mathbf{H}_{r}^{i\,*}$ and $\mathbf{b}_{r}^{i\,*}$:

$$\underbrace{\left(\sum_i \mathbf{J}_{pa}^{i\top}\mathbf{H}_{r}^{i\,*}\mathbf{J}_{pa}^{i}\right)}_{\mathbf{H}^{*}}\mathbf{x} = \underbrace{\sum_i \mathbf{J}_{pa}^{i\top}\mathbf{b}_{r}^{i\,*}}_{\mathbf{b}^{*}}, \tag{6.48}$$

(6.49)

thus yielding the same normal equation as (6.44). Note as well that Equations (6.46) and (6.42) are equivalent, thereby demonstrating that both procedures lead to the same results. In any case, the PBA amounts to the following three steps:

1. Compute relative Schur complement for every host frame i (i.e., $\mathbf{H}_{r}^{i\,*}$ and $\mathbf{b}_{r}^{i\,*}$) through Equation (6.43)).

2. Solve for pose increment $-\mathbf{x}$ using Equation (6.44)).

3. Compute point increment $-\mathbf{y}^{i}$ for all host frames i applying Equation (6.46)).

6.2.2. Relative-Absolute Pose Jacobians

In this section we present the form of the two non-zero entries of the $6 \times n$ Jacobian $\mathbf{J}_{pa}^{ij}$ (cf. Equation (6.32)), namely $\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wh}}$ and $\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wt}}$, where h, t and w refer to the host frame, the target frame and the world frame, respectively. To simplify notation, we introduce the Exp-operator

$$\mathrm{Exp} : \mathbb{R}^6 \rightarrow \mathrm{SE}(3), \tag{6.50}$$

which is a composition of the hat operator and the exponential map (cf. Section 3.4.2), i.e.:

$$\mathrm{Exp}(\boldsymbol{\xi}) = \exp(\widehat{\boldsymbol{\xi}}). \tag{6.51}$$

Its inverse operator can be defined as

$$\mathrm{Log} : \mathrm{SE}(3) \rightarrow \mathbb{R}^3. \tag{6.52}$$

Additionally to the $\oplus$-operator

$$\mathbf{T} \oplus \boldsymbol{\xi} := \exp(\widehat{\boldsymbol{\xi}})\mathbf{T}, \tag{6.53}$$

which was defined already in Equation (5.8), we introduce now the $\ominus$-operator, which allows to compute the difference between two rigid-body transformations

$$\mathbf{T}_1 \ominus \mathbf{T}_2 = \mathrm{Log}(\mathbf{T}_1 \mathbf{T}_2^{-1}), \tag{6.54}$$

such that

$$(\mathbf{T} \oplus \boldsymbol{\xi}) \ominus \mathbf{T} = \boldsymbol{\xi}. \tag{6.55}$$

Let us now define function $\mathbf{f}(\mathbf{T})$ as

$$\mathbf{f}(\mathbf{T}) : \mathrm{SE}(3) \rightarrow \mathrm{SE}(3), \tag{6.56}$$

where both its input parameter and the computed output are $\mathrm{SE}(3)$ elements. We can compute its Jacobian from the following definition of derivative:

$$\mathbf{J}_{\mathbf{f}(\mathbf{T})} = \frac{\partial \mathbf{f}(\mathbf{T})}{\partial \mathbf{T}} = \lim_{\boldsymbol{\xi} \rightarrow \mathbf{0}} \frac{\mathbf{f}(T \oplus \boldsymbol{\xi}) \ominus \mathbf{f}(\mathbf{T})}{\boldsymbol{\xi}}. \tag{6.57}$$

The computation of the relative pose between two rigid-body frames is precisely a function $\mathbf{f}$ that takes as input the absolute pose of two rigid-body frames and outputs the relative pose between them, i.e.:

$$\mathbf{T}_{th} \triangleq \mathbf{T}_{c_t c_h} = \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t}) = \mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \mathbf{T}_{wi_h} \mathbf{T}_{i_h c_h}, \tag{6.58}$$

where the subscript i refers here to a frame attached to the Inertial Measurement Unit (IMU) (which typically serves as the common reference frame in a multi-sensor (possibly multi-camera) setup), w to the world's frame and c to the camera's frame. This general framework in which we have introduced an IMU can easily be particularized to the non-IMU case simply by assuming an identity transformation between the IMU frame and the camera frame, i.e., $\mathbf{T}_{ic} = \mathbf{I}$.

Computing now $\dfrac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wh}}$ and $\dfrac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wt}}$ amounts to plugging Equation (6.58) into the definition of Jacobian presented in (6.57) for the host and target frames, respectively, giving

$$\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wh}} = \mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_h})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_h}} = \mathrm{Ad}(\mathbf{T}_{c_t i_h})\,\mathrm{Ad}(\mathbf{T}_{i_h w}), \tag{6.59}$$

$$\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wt}} = \mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_t})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_t}} = -\,\mathrm{Ad}(\mathbf{T}_{c_t i_t})\,\mathrm{Ad}(\mathbf{T}_{i_t w}). \tag{6.60}$$

In Appendix A we derive the above expressions in detail.

6.2.3. Robustification Through Iteratively Reweighted Least Squares

When building the LCW of active keyframes we only employ geometric criteria to determine covisibility constraints. Photo-consistency is not taken into account at this stage, and therefore outlier observations might still exist. The source of such outliers can be occlusions, dynamic objects moving in the viewed scene, illumination changes, or simply sensor noise.

An interesting option to dealing with outliers is modeling their root causes by estimating their parameters, e.g., tracking dynamic objects in the scene and estimating their motion. Alternatively, we can suppress outliers and their effect, e.g., by choosing different probability distributions for the residuals $p(\mathbf{r} \mid \boldsymbol{\xi})$.

In Section 3.5.3 we presented the weighted least squares method as a means to robustify the normal least squares problem against spurious measurements. (In a standard least squares framework, outliers can impact negatively on the solution, since they produce large residuals that have a high influence on the estimate through the quadratic error term.)

Recall that optimizing (6.23) is equivalent to minimizing the negative log-likelihood of the residuals given the model parameters assuming i.i.d. residuals (cf. Section 3.5.2), i.e.:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \sum_{k}^{m} \log p(r_i \mid \boldsymbol{\xi}). \tag{6.61}$$

Equating the derivatives of (6.61) to zero provides the solution $\boldsymbol{\xi}^*$, which is equivalent to the reweighted least squares problem (cf. Section 3.5.3) presented in (6.23) using weights

$$w(r_i) = -\frac{\partial \log p(r_i)}{\partial r_i} \frac{1}{r_i}, \tag{6.62}$$

if we perform the approximation of treating such weights as constant in every iteration.

Note how the residual distribution $p(r_i)$ directly affects the solution of our weighted least-squares problem. In [30] different error distributions are studied. Next, we briefly present the weighting functions corresponding to a Gaussian and a t-distribution, as well for the case of employing the Huber estimator as a robust m-estimator.

Gaussian Distribution

When assuming residuals to be normally distributed around zero, i.e., $\mathcal{N}(0, \sigma_n^2)$, then

$$p(r_i) \propto \exp(\frac{r_i^2}{\sigma_n^2}), \tag{6.63}$$

which means that all residuals are weighted equally with

$$w_n(r_i) = \frac{1}{\sigma_n^2}, \tag{6.64}$$

thus leading to a standard least squares problem.

Student's t-Distribution

The t-distribution has been proposed by several authors for robust data fitting [37, 21] and has been extensively employed in computer vision problems [18, 22, 40]. The Probability Density Function (PDF) of the t-distribution is defined as

$$p(r_i \mid \sigma_t, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu\sigma^2}} \left(1 + \frac{1}{\nu}\frac{(r_i)^2}{\sigma_t^2}\right)^{-\frac{\nu+1}{2}}, \tag{6.65}$$

where we have assumed zero-centered residuals, i.e., the mean $\mu = 0$. Moreover, σ_t is a scaling parameter, ν the degrees of freedom [4] and $\Gamma(x) = (x-1)!$ the gamma function.

Note that for $\nu \to \inf$ the t-distribution approaches a normal distribution. Indeed, we can interpret the t-distribution as an infinite mixture of Gaussian distributions [4]. As described in [30], this is a more flexible model than the i.i.d. assumption when dealing with least squares problems, where now residuals are assumed to originate from independent but non-identical distributions. In such a case, the variance cannot be dropped from Equation (3.90) and every residual is weighted with its inverse variance. Estimating the variance for every residual is intractable, but we can obtain a feasible abstraction through the t-distribution, whose weighting function is given as

$$w_t(r_i) = \frac{\nu + 1}{\nu + (\frac{r_i}{\sigma_t})^2}, \tag{6.66}$$

where we have assumed the mean μ to be zero. In previous works, the value of the degrees of freedom has been experimentally set to $\nu = 5$, first in [32] and later in [64]. In contrast to the normal distribution, the t-distribution quickly drops the weights as residuals move to the tails, thus assigning lower weights to outliers.

Fitting the t-distribution to the residuals amounts to computing the scale, which we estimate based on the current residuals using

$$\sigma_t^2 = \frac{1}{m} \sum_i^m r_i^2 \frac{\nu + 1}{\nu + (\frac{r_i^2}{\sigma_t^2})}. \tag{6.67}$$

The above equation is recursive and therefore has to be solved iteratively, but converges in a few iterations, typically around five.

Before actually fitting the t-distribution, one can filter out the gross outliers by computing an approximate initial scale $\hat{\sigma}$ through the MAD (cf. Equation (3.98)) and rejecting those residuals i for which

$$r_i > 3\hat{\sigma}. \tag{6.68}$$

Huber Estimator

The Huber estimator is a popular m-estimator that decreases the influence of high-error measurements without completely removing their influence, which allows to deal with reobservations. Small residuals are given quadratic influence, while large residuals are only weighted linearly. The Huber loss, with loss function

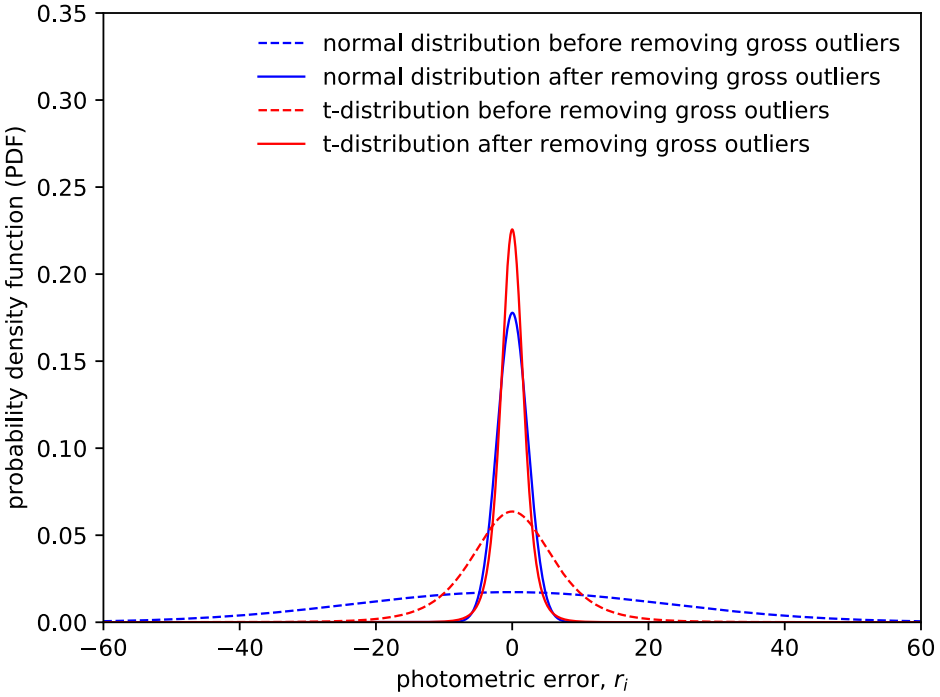


Figure 6.6.: Probabilistic error modeling. The t-distribution explains the sparse photometric model better than the normal distribution. Moreover, removing gross outliers before actually fitting the distribution is key in both cases.

$$\rho_{a_i}(r) = \begin{cases} \frac{1}{2}r^2, & \text{if } r \leq a_i \\ a_i\, r - \frac{1}{2}a_i^2, & \text{otherwise} \end{cases} \tag{6.69}$$

is convex (in contrast to other functions, e.g., the Tukey function) and therefore does not introduce new local minima to the optimization [30]. Its weight function is given by

$$w_h(r_i) = \begin{cases} \frac{1}{\sigma_n^2} & if\ |r_i| < \lambda \\ \frac{\lambda}{\sigma_n^2 |r_i|} & \text{otherwise} \end{cases}, \tag{6.70}$$

where λ can be fixed or dynamically modified each time step. For a normal distribution $\mathcal{N}(0, \sigma_n^2)$ we have $\lambda = 1.345\sigma_n$ [63], in which case outliers are given linear influence.

Probabilistic Error Modeling for the Direct Sparse Model

In [30] it is concluded that the distribution of dense photometric residuals is better explained by the t-distribution than by a normal distribution. Recently, this insight is also derived for the sparse model in [64]. In both works it is described how the t-distribution allows to quickly drop the weights assigned to photometric residuals as they move towards the tails, thus down-weighting the influence of outlier observations. Figure 6.6 shows a histogram of photometric residuals to which we have fitted both a normal (blue) and a t-distribution (red). For each case, we have fitted the corresponding distribution before and after removing gross outliers (cf. Equation (6.68)), where a dashed line is employed to represent the former cases. Note how the t-distribution better fits the residual distribution compared to the normal distribution, both before and after removing gross outliers.

6.3. Outlier Management

The stability of the PBA is largely dependent on the amount of outlier observations. Detecting and removing such observations can be achieved differently. As in [64], we exploit the fact that an observation is made up of a number of different values, as many as the number of pixels of the selected pattern. As mentioned above, we employ an eight-pixel patch as originally presented in DSO (cf. Figure 6.5). Each pixel within the patch has an associated mask, which indicates whether the pixel is inlier or outlier. A pixel observation r_i is considered an inlier if its photometric error is lower than the 95^{th} percentile of the error distribution of the target keyframe. This approach allows to seamlessly have a lower threshold for challenging keyframes, thus being more permissive, and a higher threshold for those less challenging.

Within this framework, an observation will be marked as outlier if more than 30% of the pixels in the patch are marked as outliers, as proposed in [64], in which case the observation is removed from the list of observations of the landmark.

The detection of outlier map points is closely related to the number of observations of that landmark by other keyframes in the map. Landmarks can be either *mature* or *immature*, where mature points are those that have been observed in all three consecutive keyframes after their upgrading from candidate point to landmark. On the one hand, if an immature point is not observed in all three consecutive keyframes after its creation, it is discarded forever. On the other hand, a mature landmark for which the number of observations falls below three will also be removed from the map.

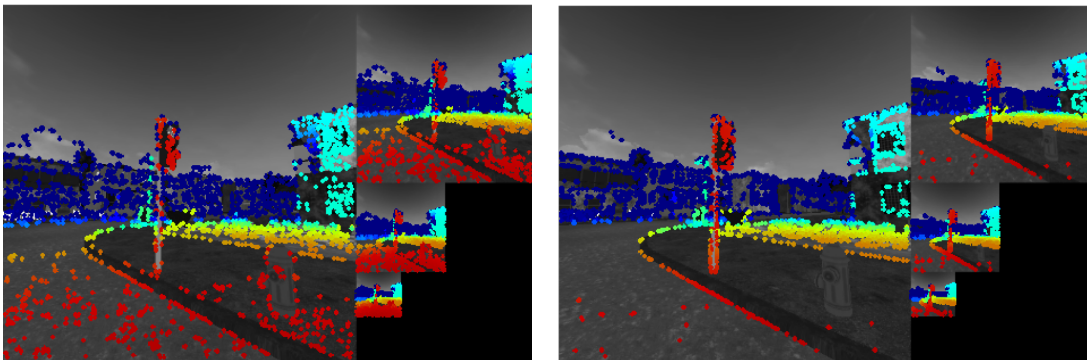
7. Evaluation

In this section we first evaluate our system's main components, namely candidate point selection, DIA (camera tracking), candidate point tracking and PBA. As test sequence we use a realistic synthetic sequence recorded using CARLA [11], an open-source game-engine-based simulator for autonomous driving research that provides both pose and structure groundtruth. The sequence comprises two loops around a roundabout in a city-like environment. As a final experiment, we run the full system on this sequence and report the obtained results, both qualitatively and quantitatively.

7.1. Candidate Point Selection and its Influence on Direct Image Alignment

Selecting good candidate points in a keyframe directly affects the chances of finding a correct match during the epipolar curve search (cf. Section 6.1.2). Indeed, if a candidate point is not distinctive enough, searching for its best match in subsequently tracked frames will typically produce poor results. On the other hand, being too restrictive when selecting high-gradient candidate points will typically leave large areas of the image empty. In general, this has a negative influence on the quality of camera tracking. Next, we describe an experiment in which we can observe this behavior. The goal is to compare the resulting Relative Pose Error (RPE) [57] between the tracking reference and the camera to be tracked under two different initial conditions, which we refer to as setting A and B, differing solely in the set of points used to create the tracking reference. (Note, however, that the number of selected candidate points is $N_{cps} = 2000$ in both cases, as described in Section 6.1.1.)

The general setup for both scenarios is as follows. Given an initial keyframe whose pose at time t is known, as well as its intensity image and its groundtruth depth map (let us assume for this experiment that the structure, i.e., depth, is known, rather than having to estimate it in subsequent frames as was explained in Section 6.1.2), we want to estimate the pose of a subsequent frame whose intensity image is also given. (See Section 5.1 for a more detailed description of camera tracking.) The pose of this new frame is initialized with that of the keyframe. We then select candidate points (image locations) on the keyframe's intensity image and convert the groundtruth depth associated to each selected pixel into an inverse distance value, thus generating a sparse inverse distance map, such as those shown in Figure 7.1. Under such setup we can now run either one of the algorithms presented for DIA in Section 5.1 (for this experiment we use the FA approach) and estimate the relative pose of the new frame with respect to the keyframe.



(a) Setting A **(b)** Setting B

Figure 7.1.: Influence of candidate point selection on camera tracking. Using a gradient-based threshold for a given region i of $\bar{g}_i + g_{th}$ allows us to select points that are both well-distributed across the image and with a sufficiently high gradient magnitude relative to the surrounding pixels (a). On the other hand, employing a threshold four times greater (b), and therefore being more strict, produces an unequal distribution of points, which affects the quality of camera tracking.

Tracking accuracy, nevertheless, depends on the set of selected points with which we create the tracking reference, which is shown in Table 7.1 for the two different settings considered: in A we employ a gradient-based threshold of $\bar{g}_i + g_{th}$ (where i is a given region in the image as described in Section 6.1.1). This results in an RPE of 0.2 mm, which is nine times smaller than that obtained with a threshold four times higher, i.e., $4(\bar{g}_i + g_{th})$, which corresponds to setting B. Figure 7.1 shows the generated tracking references for each setting, where it can be observed that in the setting A (Figure 7.1a) points are well-distributed across the image, whereas for the setting B (Figure 7.1b) the concentration of points is too high in some areas but insufficient in others (e.g., the lower half of the image).

A possible explanation as to why we obtain worse tracking results in setting B resides in the fact that the keyframe's intensity image (i.e., the underlying grayscale image in either one of the images of Figure 7.1) shows an intensity constancy along the image diagonal (essentially, the intersection line between the road and the sidewalk). By selecting candidate points mainly on such line, motion is left slightly unconstrained along this direction. If the camera then moves

	Gradient-based threshold	RPE (mm)
Before tracking	-	143
Setting A	$\bar{g}_i + g_{th}$	0.2
Setting B	$4(\bar{g}_i + g_{th})$	1.8

Table 7.1.: Influence of candidate point selection on the RPE after Direct Image Alignment. The number of selected points is 2000 in both cases. However, for the setting B the region-threshold employed during candidate point selection is set to four times that employed in the setting A, which results in worse tracking results. In particular, the resulting RPE for B is nine times higher compared to that produced under the initial conditions of A. Nevertheless, note that in absolute terms both settings allow to notably reduce the RPE compared to the initial value.

along such unconstrained direction, as it is our case, the pose estimate will be affected negatively. Therefore, as it was already indicated by Engel et al. in DSO [13], selecting points in relatively weakly-textured regions (e.g., bottom-left part on the intensity image in Figure 7.1), camera motion is in general more constrained.

It should be noted, however, that regardless of the set of candidate points selected, a coarse-to-fine scheme has to be employed to account for the large displacement. Otherwise, tracking fails altogether in both cases. Additionally, employing a robust error function based on the fitting of the t-distribution to the residuals (cf. Section 6.2.3) helps refine the estimate by neutralizing the influence of outliers.

7.2. Comparison of the Forward Additive and the Inverse Compositional Algorithms for Direct Image Alignment

We have compared our implementations for the FA and IC approaches (cf. Section 5.1). The results are summarized in Table 7.2. For each algorithm we have a non-parallelized (FAnon-p and ICnon-p) and a parallelized version (FAp and ICp). In the experiment we have run each version five times. The RPE is consistently the same across runs for a given approach (regardless of whether it uses the parallelized mode or not). Runtimes, on the other hand, are averaged over all five runs for each of the four cases. Moreover, note that the number of iterations per level (shown in parenthesis in Table 7.2) is also the same across runs.

We have used the same setup as that described in the previous section, with candidate points being selected using a gradient-based threshold of $\bar{g}_i + g_{th}$ (denoted as setting A in the previous section), so as to achieve an equal distribution of points in the image. We use 3 additional pyramid levels (where 0 is the index of the finest level), and allow a maximum of 20 G-N iterations per level. For each approach, namely FA and IC, we report the final RPE between the keyframe and the tracked frame. (Note that parallelization does not affect the resulting RPE but the runtime.) We observe that, in this particular case, IC achieves a lower RPE than FA. We have not observed, however, that this is a fixed trend in general.

In terms of runtime, both parallelized versions (i.e., FAp and ICp) largely outperform their non-parallelized counterparts. Moreover, we see that FA outperforms IC in terms of overall runtime and runtime per iteration (see column **i** in Table 7.2). While in theory the IC formulation should be faster (given that it can precompute the Jacobians at the beginning of every level and reuse them at every iteration of the same level), in practice accessing the cached Jacobians increases the overall runtime per iteration in our implementation. Nevertheless, the comparing the *total* FAp vs. ICp runtime (or the FAnon-p vs. ICnon-p runtime) should be done with caution, since the total number of iterations per level differs between approaches. In this particular example, while FA carries out 5 iterations in the coarsest level, IC performs a total of 13 in the same level.

Approach	RPE (mm)	Metric								
		Runtime (ms)								
		Total (ms)	Per Level							
			3		2		1		0	
			p	i	p	i	p	i	p	i
FAnon-p	0.2	115.28	-	0.3 (5)	-	0.9 (20)	-	2 (5)	-	3.7 (20)
FAp		37.6	-	0.1 (5)	-	0.15 (20)	-	0.2 (5)	-	0.3 (20)
ICnon-p	0.14	166.1	0.7	0.37 (13)	1.3	1.1 (20)	3.8	2.5 (4)	7.2	4 (20)
ICp		56.4		0.16 (13)		0.4 (20)		0.7 (4)		1.2 (20)

Table 7.2.: Comparison of the FA and IC approaches for DIA. For each approach and mode (non-parallelized vs. parallelized) we report the mean value of five runs. We use three additional pyramid levels (with 0 being the index of the finest level), and allow a maximum of 20 iterations per level. In the IC approach we precompute the Jacobians at the beginning of every level, since they are constant across iterations. We report the precomputation runtime for each level under the column **p**. For FA this precomputation runtime is zero, since Jacobians need to be recomputed at every iteration. Columns **i** report the average runtime per iteration, which of course differs at each level for a given variant. In parenthesis we show the number of iterations performed in the corresponding level. Both the RPE and the number of iterations per level have to be necessarily the same in the non-parallelized and parallelized modes, given that the underlying problem being solved does not change.

7.3. Qualitative Evaluation of Candidate Point Tracking

As presented in Section 6.1.2, we use newly tracked frames to refine the inverse distance estimate of every candidate point hosted in each one of the temporal keyframe. In Figure 6.4 we showed how candidate points are refined across several subsequent frames. To facilitate reading we present the same example again in Figure 7.2. In this particular case, the third frame is additionally converted into a keyframe. Therefore, the LCW is recomputed and new candidate points are upgraded to landmarks (purple points on the bottom part of Figure 7.2). Candidate points for which a good inverse distance estimate is not yet available are kept as candidate points; their inverse distance will continue to be refined across new tracked frames and, if good enough, might be upgraded to landmarks at a later point.

Figure 7.3 visualizes another example of how candidate points are refined across subsequent frames. Note that, at $t + 1$ (i.e., after having used the first subsequent tracked frame to trace candidate points), the point cloud of candidate points for the keyframe in this example does not present a clear structure. However, at $t + 3$ (i.e., after having traced the candidate points across three subsequent frames), some structures start to become visible, such as the plane within the circle A.

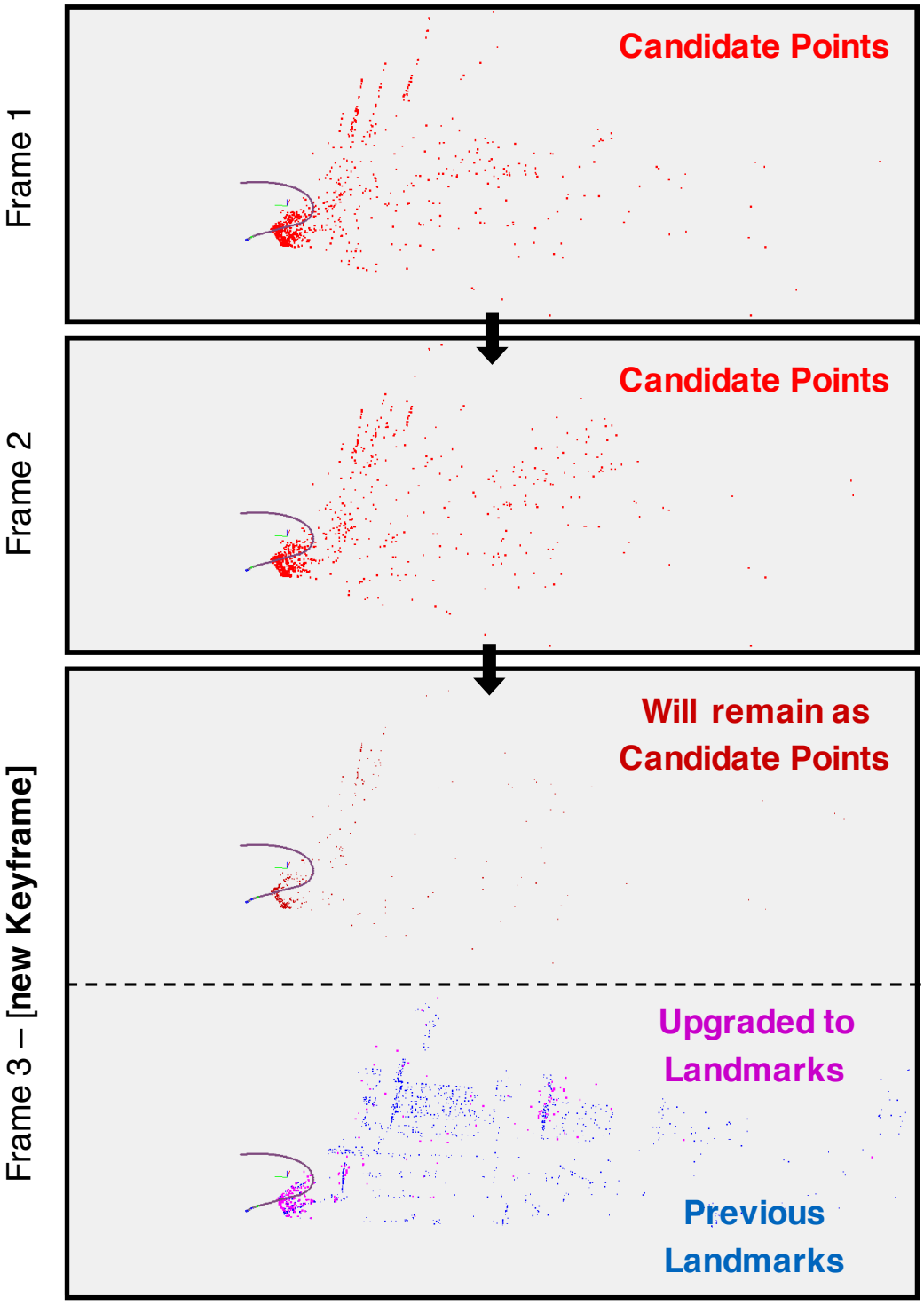


Figure 7.2.: Candidate point tracking and their upgrading to landmark.

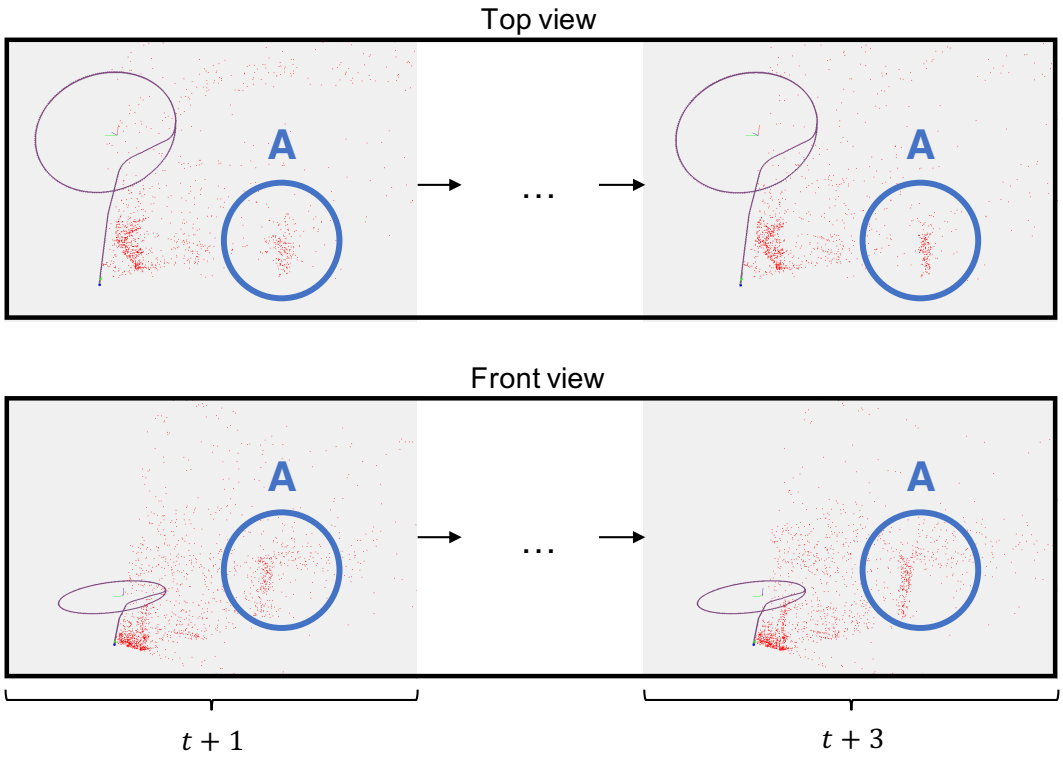


Figure 7.3.: Example of the refinement of structure throughout subsequent operations of candidate point tracking.

7.4. Qualitative and Quantitative Evaluation of our Solver Implementations for Photometric Bundle Adjustment

We evaluate our custom solver for the PBA problem on a small test map with 12 cameras, which host around 3000 landmarks in total. The map was obtained as follows. First, we ran DSO on our synthetic test sequence. Second, we extracted 6 cameras from a first pass over a certain region and another 6 from the second pass over the same area. This test map can be visualized in Figure 7.4. Finally, we computed all observations for each landmark based on geometric constraints (i.e., whether a landmark reprojects or not into another frame), without any consideration about photometric consistency between observations.

On this test map we compare our custom solver for the PBA problem versus our implementation using the Ceres framework [1]. We do this for the non pyramidal approach (i.e., w/o employing a coarse-to-fine scheme). Moreover, we also evaluate our custom solver w/ a coarse-to-fine scheme. (Note that we do not have implemented a Ceres version w/ pyramid levels). Table 7.3 shows the resulting Absolute Trajectory Error (ATE) [57] as well as the Total Runtime (TR) and Average Runtime per Iteration (ARI) when allowing a maximum of 10 iterations per level. Table 7.4 shows these same metrics for the case in which we allow a maximum of 50 iterations per level.

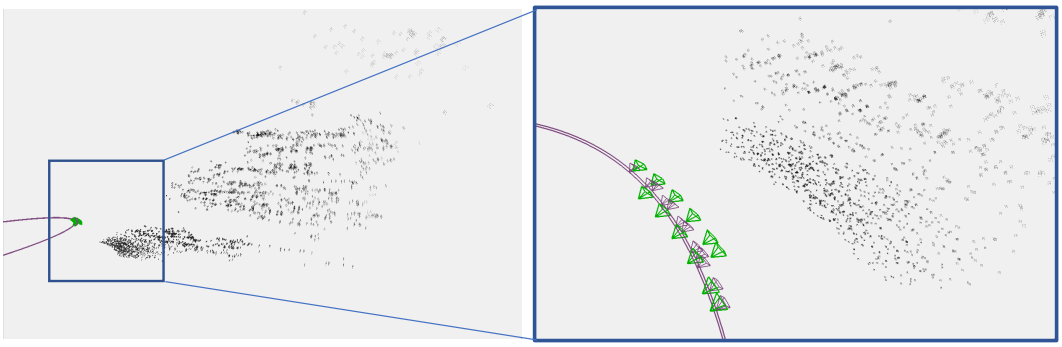


Figure 7.4.: Test map before PBA. Out of the total 12 cameras (green pyramids), 6 correspond to the first pass of the sensor across the region, while the other half belong to a second pass of the sensor over the same area. The purple pyramids represent groundtruth poses, while each of the two purple lines interpolates the groundtruth trajectory of the sensor for a given pass over the visualized section of the scene. Black dots correspond to all landmarks hosted in the 12 cameras. Note how current poses (green pyramids) do not align with groundtruth (purple pyramids).

Solver Type	Metric							
	ATE (m)	TR (s)	ARI (s)	Iterations				
				Total	Level			
					3	2	1	0
Manual w/o pyrs	0.00131	0.43	0.043	10	-	-	-	10
Ceres w/o pyrs	0.03761	2.05	0.20	10	-	-	-	10
Manual w/ pyrs	0.00054	2.21	0.076	29	7	3	10	9
Ceres w/ pyrs	-	-	-	-	-	-	-	-

Table 7.3.: Comparison between different solvers for the PBA problem when allowing a maximum of 10 iterations per level. TR stands for Total Runtime, whereas ARI denotes Average Runtime per Iteration. Both are measured in seconds. The latter is computed by dividing TR between the total number of iterations.

Solver Type	Metric							
	ATE (m)	TR (s)	ARI (s)	Iterations				
				Total	Level			
					3	2	1	0
Manual w/o pyrs	0.00059	1.21	0.052	23	-	-	-	23
Ceres w/o pyrs	0.00071	8.23	0.165	50	-	-	-	50
Manual w/ pyrs	0.00054	2.53	0.081	31	7	3	12	9
Ceres w/ pyrs	-	-	-	-	-	-	-	-

Table 7.4.: Comparison between different solvers for the PBA problem when allowing a maximum of 50 iterations per level. TR stands for Total Runtime, whereas ARI denotes Average Runtime per Iteration. Both are measured in seconds. The latter is computed by dividing TR between the total number of iterations.

For the non-pyramidal approach, note how our manual solver compares positively in terms of runtime versus our implementation in Ceres. When allowing a maximum of 10 iterations per level, our custom solver is around 5 times faster (0.2 / 0.043), which we can obtain from comparing the ARI in each case. On the other hand, when allowing a maximum of 50 iterations, it is around 3 times faster (0.165 / 0.052). In any case, it is clear that our custom solver is faster than our solver implementation using Ceres, which was to some extent expected, given the fact that when implementing a solver manually for a particular task (in this case PBA), many optimizations can be performed that are particular to the given problem. Realizing such optimizations in a general-purpose solver, such as Ceres, is typically more complex and sometimes not even possible. Finally, with respect to the final ATE, our custom solver also beats our Ceres implementation, for both the 10- and 50-iteration scenarios. This result should be treated with caution, the reason being that in both solver types we use the L-M algorithm, whose behavior depends on the parameter λ in Equation (3.83), and on how much it is increased or decreased across iterations.

When comparing our custom solver w/ and w/o a coarse-to-fine scheme, we observe that a lower ATE is achieved by the former, but at the expense of a higher runtime. While in theory the ARI should be the same for both variations (i.e., w/ and w/o a pyramidal approach), in practice we see that they differ slightly, which might be caused by the overhead of setting up the problem at every new level.

Figure 7.5 shows how the resulting map (cameras and points) is refined after running our custom PBA solver with a coarse-to-fine scheme for a maximum of 10 iterations per level. Regarding camera poses, these get closer to the groundtruth poses (purple), thus the reduction in the ATE shown in Table 7.3. Measuring the improvement for points is more difficult to quantify. Visually, however, it can be observed how points converge into planar structures, thus resulting in a more clear geometry.

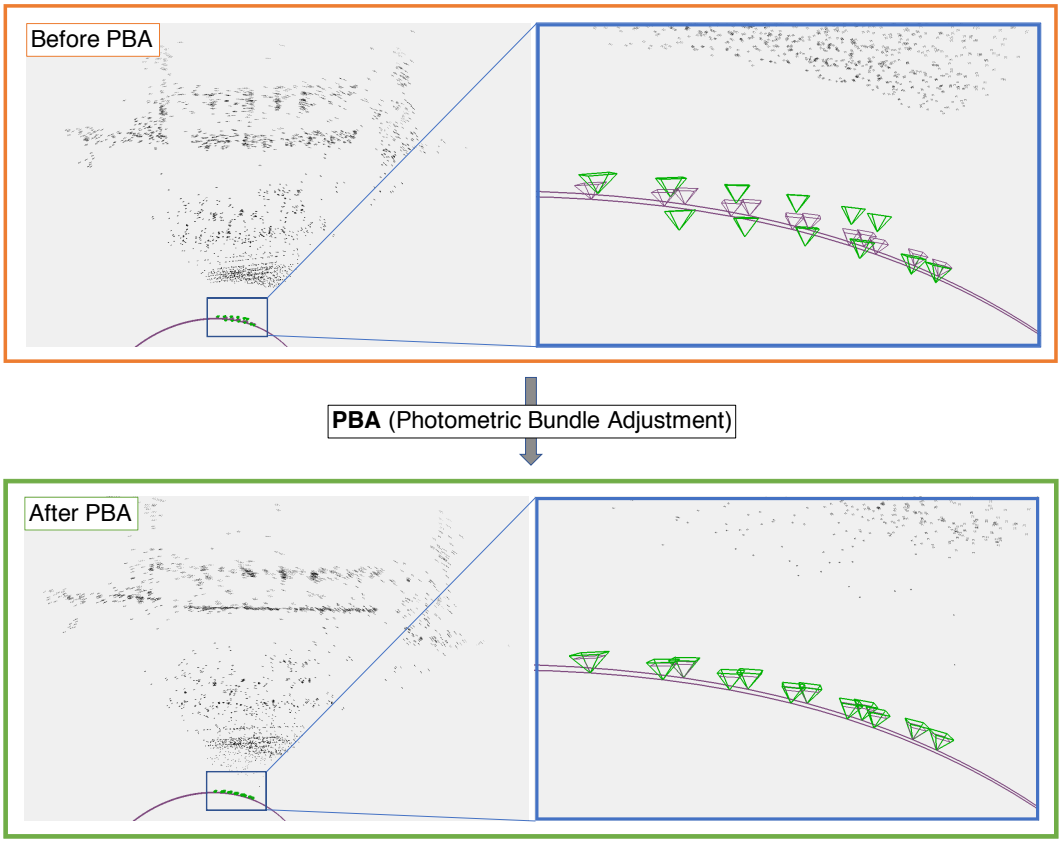


Figure 7.5.: Example of PBA, where cameras and points are jointly optimized, resulting in cameras being closer to their groundtruth pose and points better representing the world's geometry.

7.5. Qualitative and Quantitative Evaluation of the Direct SLAM Method on a Synthetic Sequence

We have evaluated our direct SLAM on our synthetic test sequence, which comprises two loops around a roundabout in a city-like environment. The full reconstructed scene is shown in Figure 7.7, whereas Figure 7.8 shows the reconstructed map after a partial sequence (out of the complete test sequence). Table 7.5 shows the resulting ATE for each sequence, namely 1.05 m for the complete sequence, which comprises 314 frames in total, and 0.005 m for the partial sequence, made up of 100 frames. We observe that our proposed method is not able to deal with large loops, as can be concluded by looking at the accumulated drift of the trajectory followed by the camera in Figure 7.7. The intuition behind this finding is that the LCW is only able to deal with soft loop closures, given that the PBA's radius of convergence is relatively small. Therefore, the system fails to correct the accumulated error if the camera revisits already mapped areas with high drift.

However, for a shorter sequence without loops, such as that shown in Figure 7.8, the final ATE is relatively small (around 5 mm). In the middle and bottom views of this figure, it can be

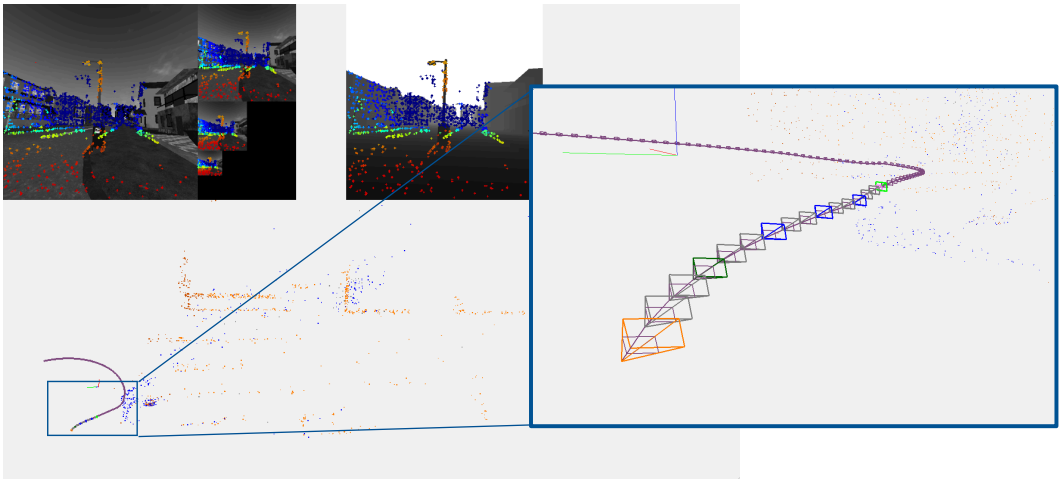


Figure 7.6.: Example of landmark reuse. Previously mapped points (orange dots) hosted in an active covisible keyframe (orange pyramid) are reactivated for the current keyframe (light green pyramid at the top right part of the close-up image). This avoids duplicating landmarks and helps to reduce drift.

	Frames	ATE (m)
Complete sequence	314	1.05
Partial sequence	100	0.005

Table 7.5.: Absolute Trajectory Error for the complete and partial sequences.

observed how active covisible and inactive covisible keyframes (light and dark orange cameras, respectively), get correctly selected based on covisibility constraints with respect to the keyframes in the temporal window (blue cameras) and the last keyframe (light green camera).

Additionally, Figure 7.6 shows how previously mapped points (orange dots) hosted in an active covisible keyframe (orange pyramid withing the close-up) can be activated and reused when needed based on covisibility constraints with the latest keyframe (light green pyramid within the close-up), thereby avoiding landmark duplication and maintaining a consistent map.

Overall, in this section we have shown that the individual modules that conform our system are correctly implemented, giving reasonable performance and accuracy. Moreover, we have deployed our direct SLAM method on a large-scale synthetic automotive sequence, thus demonstrating that the complete system is fully functional and ready to serve as a framework for future research in the field.

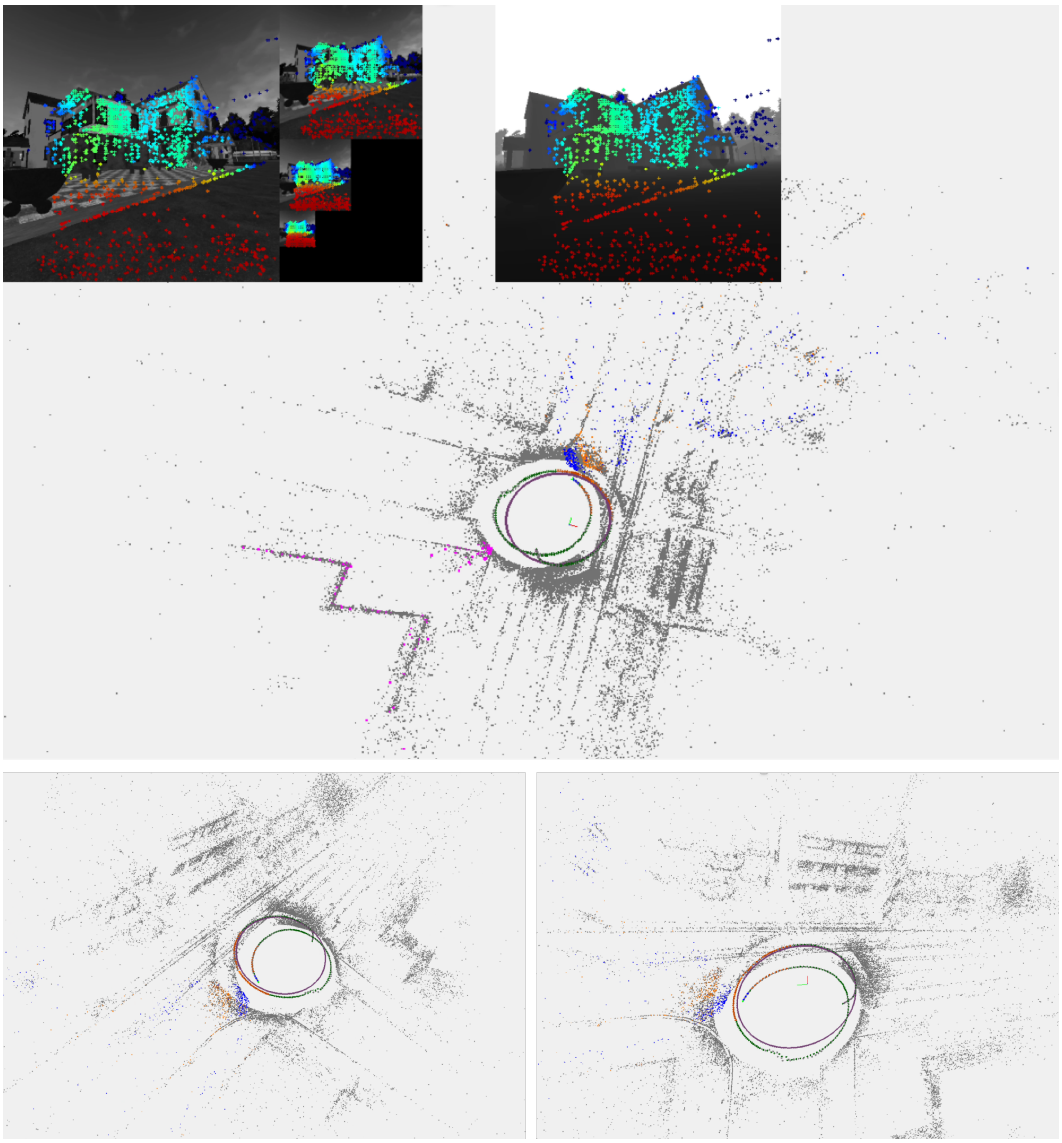


Figure 7.7.: Screenshots of a map reconstructed using our direct SLAM system on a synthetic test sequence. The camera follows a circular trajectory around a roundabout in a city-like scenario. The reconstructed camera poses closely follow groundtruth during approximately the initial 100 frames, drifting afterwards and being unable to close the loop at the end of the sequence when returning to the initial region. This relates to the fact that the radius of convergence in the PBA is relatively small, thus requiring that the initial estimate is close to the optimum.

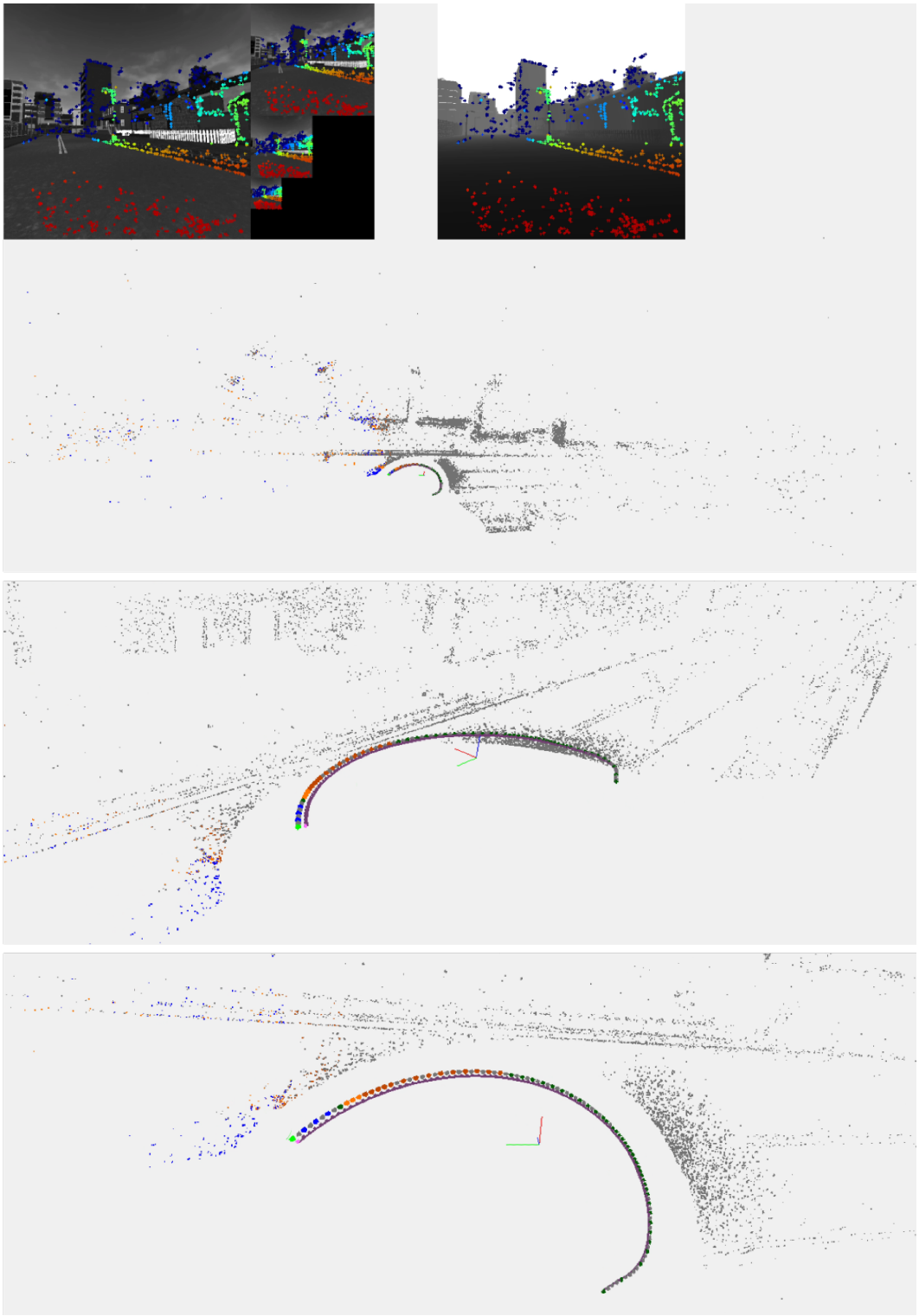


Figure 7.8.: Screenshots of a map reconstructed using our direct SLAM system on a partial sequence of the complete synthetic test sequence. Note how active covisible and inactive covisible keyframes (light and dark orange cameras, respectively), get correctly selected based on covisibility constraints with respect to the keyframes in the temporal window (blue cameras) and the last keyframe (light green).

8. Conclusion

In this thesis, we have presented a direct SLAM method that explicitly deals with map point re-observations by handling covisibility constraints between keyframes. We have implemented and described in detail all major components, including DIA (in its forward additive and inverse compositional variants), map point selection and depth initialization, keyframe and point management (in particular, the reuse of previously mapped regions to ensure a consistent representation of the scene), and local window PBA, which have been combined into a modular and flexible direct SLAM framework.

Furthermore, we have implemented a coarse-to-fine scheme and a robust weighting scheme (based on the t-distribution) to further ensure the stability of both tracking and PBA, effectively reducing the influence of spurious observations that do not follow the photo-consistency assumption, such as occlusions and disocclusions. Additionally, the proposed method is not limited to the pinhole model. Unlike other existing implementations, it has been designed to work directly on distorted images, thus being suitable for fish-eye lenses.

We have carried out an evaluation of the system on realistic synthetic data from a game-engine-based simulator with both pose and structure groundtruth. We have validated both the individual components of the system as well as its overall performance, deploying the proposed method on a large-scale synthetic automotive sequence. In doing so we have shown that the complete direct SLAM system presented here is fully functional and ready to serve as an extensible framework for future research in this field.

Future work comprises extending the current system with loop closure detection and subsequent pose graph optimization –which would re-enable the benefits of the LCW even after large loops–, as well as evaluating the method on real datasets, so as to directly compare it with state-of-the-art approaches.

A. Relative-Absolute Pose Jacobian Derivation

In Section 6.2.2 we presented the form of the two non-zero entries of the $6 \times n$ Jacobian $\mathbf{J}_{pa}^{ij}$ (cf. Equation (6.32)):

$$\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wh}} = \mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_h})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_h}} = \mathrm{Ad}(\mathbf{T}_{c_t i_h})\,\mathrm{Ad}(\mathbf{T}_{i_h w}), \tag{A.1}$$

$$\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wt}} = \mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_t})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_t}} = -\mathrm{Ad}(\mathbf{T}_{c_t i_t})\,\mathrm{Ad}(\mathbf{T}_{i_t w}), \tag{A.2}$$

where h, t and w refer to the host frame, the target frame and the world frame, respectively. In this appendix we derive the above expressions. We begin with $\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wh}}$, which can be derived as follows:

$$\mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_h})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_h}} \tag{A.3}$$

$$= \frac{\partial \mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \mathbf{T}_{wi_h} \mathbf{T}_{i_h c_h}}{\partial \mathbf{T}_{wi_h}} \tag{A.4}$$

$$= \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \operatorname{Exp}(\boldsymbol{\xi}) \mathbf{T}_{wi_h} \mathbf{T}_{i_h c_h} (\mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \mathbf{T}_{wi_h} \mathbf{T}_{i_h c_h})^{-1}\right)}{\boldsymbol{\xi}} \tag{A.5}$$

$$= \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \operatorname{Exp}(\boldsymbol{\xi}) \overbrace{\mathbf{T}_{wi_h} \mathbf{T}_{i_h c_h} \mathbf{T}_{i_h c_h}^{-1} \mathbf{T}_{wi_h}^{-1}}^{\mathbf{I}} \mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t}\right)}{\boldsymbol{\xi}} \tag{A.6}$$

$$= \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_t c_t}^{-1} \mathbf{T}_{wi_t}^{-1} \operatorname{Exp}(\boldsymbol{\xi}) \mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t}\right)}{\boldsymbol{\xi}} \tag{A.7}$$

$$= \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Log}\left((\mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t})^{-1} \operatorname{Exp}(\boldsymbol{\xi}) \mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t}\right)}{\boldsymbol{\xi}} \tag{A.8}$$

$$\overset{(3.66)}{=} \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Log}\left(\operatorname{Exp}\left(\operatorname{Ad}((\mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t})^{-1}) \boldsymbol{\xi}\right)\right)}{\boldsymbol{\xi}} \tag{A.9}$$

$$= \lim_{\boldsymbol{\xi} \to \mathbf{0}} \frac{\operatorname{Ad}((\mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t})^{-1}) \boldsymbol{\xi}}{\boldsymbol{\xi}} \tag{A.10}$$

$$= \operatorname{Ad}((\mathbf{T}_{wi_t} \mathbf{T}_{i_t c_t})^{-1}) \tag{A.11}$$

$$= \operatorname{Ad}(\mathbf{T}_{c_t i_t} \mathbf{T}_{i_t w}) \tag{A.12}$$

$$= \operatorname{Ad}(\mathbf{T}_{c_t w}) \tag{A.13}$$

$$= \operatorname{Ad}(\mathbf{T}_{c_t i_h} \mathbf{T}_{i_h w}) \tag{A.14}$$

$$= \operatorname{Ad}(\mathbf{T}_{c_t i_h}) \operatorname{Ad}(\mathbf{T}_{i_h w}). \tag{A.15}$$

Analogously, we can obtain $\frac{\partial \mathbf{T}_{th}}{\partial \mathbf{T}_{wt}}$ as follows:

$$\mathbf{J}_{\mathbf{f}(\mathbf{T}_{wi_t})} = \frac{\partial \mathbf{f}(\mathbf{T}_{wi_h}, \mathbf{T}_{wi_t})}{\partial \mathbf{T}_{wi_t}} \tag{A.16}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_tc_t}^{-1}(\operatorname{Exp}(\boldsymbol{\xi})\mathbf{T}_{wi_t})^{-1}\mathbf{T}_{wi_h}\mathbf{T}_{i_hc_h}(\mathbf{T}_{i_tc_t}^{-1}\mathbf{T}_{wi_t}^{-1}\mathbf{T}_{wi_h}\mathbf{T}_{i_hc_h})^{-1}\right)}{\boldsymbol{\xi}} \tag{A.17}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_tc_t}^{-1}\mathbf{T}_{wi_t}^{-1}\operatorname{Exp}(-\boldsymbol{\xi})\mathbf{T}_{wi_h}\mathbf{T}_{i_hc_h}\mathbf{T}_{i_hc_h}^{-1}\mathbf{T}_{wi_h}^{-1}\mathbf{T}_{wi_t}\mathbf{T}_{i_tc_t}\right)}{\boldsymbol{\xi}} \tag{A.18}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{i_tc_t}^{-1}\mathbf{T}_{wi_t}^{-1}\operatorname{Exp}(-\boldsymbol{\xi})\mathbf{T}_{wi_t}\mathbf{T}_{i_tc_t}\right)}{\boldsymbol{\xi}} \tag{A.19}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{c_ti_t}\mathbf{T}_{i_tw}\operatorname{Exp}(-\boldsymbol{\xi})\mathbf{T}_{wc_t}\right)}{\boldsymbol{\xi}} \tag{A.20}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{c_tw}\operatorname{Exp}(-\boldsymbol{\xi})\mathbf{T}_{wc_t}\right)}{\boldsymbol{\xi}} \tag{A.21}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\mathbf{T}_{c_tw}\operatorname{Exp}(-\boldsymbol{\xi})\mathbf{T}_{c_tw}^{-1}\right)}{\boldsymbol{\xi}} \tag{A.22}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\operatorname{Log}\left(\operatorname{Exp}(-\operatorname{Ad}(\mathbf{T}_{c_tw})\boldsymbol{\xi})\right)}{\boldsymbol{\xi}} \tag{A.23}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{-\operatorname{Ad}(\mathbf{T}_{c_tw})\boldsymbol{\xi}}{\boldsymbol{\xi}} \tag{A.24}$$

$$= -\operatorname{Ad}(\mathbf{T}_{c_tw}) \tag{A.25}$$

$$= -\operatorname{Ad}(\mathbf{T}_{c_ti_t})\operatorname{Ad}(\mathbf{T}_{i_tw}). \tag{A.26}$$

A.1. Derivation through the Chain Rule

Using the chain rule we can obtain the same results as in (A.15), which we briefly demonstrate. First, we have that the relative rigid-body transform mapping points from the camera frame of the target keyframe to the camera frame of the host keyframe can be written down as

$$\mathbf{T}_{rel\_cam}(\mathbf{T}_{rel\_imu}(\mathbf{T}_{abs\_h}, \mathbf{T}_{abs\_t})), \tag{A.27}$$

where

$$\mathbf{T}_{rel\_imu}(\mathbf{T}_{abs\_h}, \mathbf{T}_{abs\_t}) = \mathbf{T}_{i_ti_h} = \mathbf{T}_{wi_t}^{-1}\mathbf{T}_{wi_h}, \tag{A.28}$$

$$\mathbf{T}_{rel\_cam} = \mathbf{T}_{i_tc_t}^{-1}\mathbf{T}_{i_ti_h}\mathbf{T}_{i_hc_h}. \tag{A.29}$$

The partial derivative of $\mathbf{T}_{rel\_imu}$ (Equation (A.28)) with respect to $\mathbf{T}_{wi_h}$ can be computed applying the definition of Jacobian presented in Equation (6.57), giving

$$\frac{\partial \mathbf{T}_{rel\_imu}}{\partial \mathbf{T}_{wi_h}} = \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathbf{T}_{wi_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi})\mathbf{T}_{wi_h}(\mathbf{T}_{wi_t}^{-1}\mathbf{T}_{wi_h})^{-1}\right)}{\boldsymbol{\xi}} \tag{A.30}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathbf{T}_{wi_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi})\mathbf{T}_{wi_h}\mathbf{T}_{wi_h}^{-1}\mathbf{T}_{wi_t}\right)}{\boldsymbol{\xi}} \tag{A.31}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathbf{T}_{wi_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi})\mathbf{T}_{wi_t}\right)}{\boldsymbol{\xi}} \tag{A.32}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathrm{Exp}(\mathrm{Ad}(\mathbf{T}_{wi_t}^{-1})\boldsymbol{\xi})\right)}{\boldsymbol{\xi}} \tag{A.33}$$

$$= \lim_{\boldsymbol{\xi}\to\mathbf{0}} \frac{\mathrm{Ad}(\mathbf{T}_{wi_t}^{-1})\boldsymbol{\xi}}{\boldsymbol{\xi}} \tag{A.34}$$

$$= \mathrm{Ad}(\mathbf{T}_{wi_t}^{-1}). \tag{A.35}$$

On the other hand, the partial derivative of $\mathbf{T}_{rel\_cam}$ (Equation (A.29)) with respect to $\mathbf{T}_{rel\_imu}$ is obtained as follows:

$$\frac{\partial \mathbf{T}_{rel\_cam}}{\partial \mathbf{T}_{rel\_imu}} = \frac{\partial \mathbf{T}_{rel\_cam}}{\partial \mathbf{T}_{i_t i_h}} = \lim_{\boldsymbol{\xi}_{i_t i_h}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathbf{T}_{i_t c_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi}_{i_t i_h})\mathbf{T}_{i_t i_h}\mathbf{T}_{i_h c_h}(\mathbf{T}_{i_t c_t}^{-1}\mathbf{T}_{i_t i_h}\mathbf{T}_{i_h c_h})^{-1}\right)}{\boldsymbol{\xi}} \tag{A.36}$$

$$= \lim_{\boldsymbol{\xi}_{i_t i_h}\to\mathbf{0}} \frac{\mathrm{Log}\left(T_{i_t c_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi}_{i_t i_h})\mathbf{T}_{i_t i_h}\mathbf{T}_{i_h c_h}\mathbf{T}_{i_h c_h}^{-1}\mathbf{T}_{i_t i_h}^{-1}\mathbf{T}_{i_t c_t}\right)}{\boldsymbol{\xi}} \tag{A.37}$$

$$= \lim_{\boldsymbol{\xi}_{i_t i_h}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathbf{T}_{i_t c_t}^{-1}\,\mathrm{Exp}(\boldsymbol{\xi}_{i_t i_h})\mathbf{T}_{i_t c_t}\right)}{\boldsymbol{\xi}} \tag{A.38}$$

$$= \lim_{\boldsymbol{\xi}_{i_t i_h}\to\mathbf{0}} \frac{\mathrm{Log}\left(\mathrm{Exp}(\mathrm{Ad}(\mathbf{T}_{i_t c_t}^{-1})\boldsymbol{\xi}_{i_t i_h})\right)}{\boldsymbol{\xi}} \tag{A.39}$$

$$= \lim_{\boldsymbol{\xi}_{i_t i_h}\to\mathbf{0}} \frac{\mathrm{Ad}(\mathbf{T}_{i_t c_t}^{-1})\boldsymbol{\xi}_{i_t i_h}}{\boldsymbol{\xi}} \tag{A.40}$$

$$= \mathrm{Ad}(\mathbf{T}_{i_t c_t}^{-1}). \tag{A.41}$$

Combining Equations (A.35) and (A.41) finally gives

$$\frac{\partial \mathbf{T}_{rel\_cam}}{\partial \mathbf{T}_{wi_h}} = \frac{\partial \mathbf{T}_{rel\_cam}}{\partial \mathbf{T}_{rel\_imu}} \frac{\partial \mathbf{T}_{rel\_imu}}{\partial \mathbf{T}_{wi_h}} \tag{A.42}$$

$$= \mathrm{Ad}(\mathbf{T}_{i_t c_t}^{-1})\,\mathrm{Ad}(\mathbf{T}_{wi_t}^{-1}) \tag{A.43}$$

$$= \mathrm{Ad}(\mathbf{T}_{c_t i_t})\,\mathrm{Ad}(\mathbf{T}_{i_t w}) \tag{A.44}$$

$$= \mathrm{Ad}(\mathbf{T}_{c_t w}) \tag{A.45}$$

$$= \mathrm{Ad}(\mathbf{T}_{c_t i_h}\mathbf{T}_{i_h w}) \tag{A.46}$$

$$= \mathrm{Ad}(\mathbf{T}_{c_t i_h})\,\mathrm{Ad}(\mathbf{T}_{i_h w}), \tag{A.47}$$

which is equivalent to (A.15), demonstrating how the chain rule can be applied in such cases.

Bibliography

[1] S. Agarwal, K. Mierle, et al. *Ceres Solver*. http://ceres-solver.org.

[2] S. Baker and I. Matthews. “Equivalence and efficiency of image alignment algorithms”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. Citeseer. 2001, pp. I–1090.

[3] S. Baker and I. Matthews. “Lucas-kanade 20 years on: A unifying framework”. In: *International journal of computer vision* 56.3 (2004), pp. 221–255.

[4] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] D. Caruso, J. Engel, and D. Cremers. “Large-scale direct slam for omnidirectional cameras”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 141–148.

[6] J. Civera, A. J. Davison, and J. M. Montiel. “Inverse depth parametrization for monocular SLAM”. In: *IEEE transactions on robotics* 24.5 (2008), pp. 932–945.

[7] A. I. Comport, E. Malis, and P. Rives. “Accurate quadrifocal tracking for robust 3d visual odometry”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 40–45.

[8] A. I. Comport, E. Malis, and P. Rives. “Real-time quadrifocal visual odometry”. In: *The International Journal of Robotics Research* 29.2-3 (2010), pp. 245–266.

[9] A. Concha and J. Civera. “Using superpixels in monocular SLAM”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 365–372.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2007), pp. 1052–1067.

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

[12] E. Eade and T. Drummond. “Edge Landmarks in Monocular SLAM.” In: *BMVC*. 2006, pp. 7–16.

[13] J. Engel, V. Koltun, and D. Cremers. “Direct sparse odometry”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[14] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[15] J. Engel, J. Sturm, and D. Cremers. “Semi-dense visual odometry for a monocular camera”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1449–1456.

[16] M. A. Fischler and R. C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 15–22.

[18] J. Gai and R. L. Stevenson. “Studentized dynamical system for robust object tracking”. In: *IEEE Transactions on Image Processing* 20.1 (2010), pp. 186–199.

[19] D. Gálvez-López and J. D. Tardos. “Bags of binary words for fast place recognition in image sequences”. In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[20] X. Gao, R. Wang, N. Demmel, and D. Cremers. “LDSO: Direct sparse odometry with loop closure”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2198–2204.

[21] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[22] D. Gerogiannis, C. Nikou, and A. Likas. “Robust Image Registration using Mixtures of t-distributions”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[23] C. Geyer and K. Daniilidis. “A unifying theory for central panoramic systems and practical implications”. In: *European conference on computer vision*. Springer. 2000, pp. 445–461.

[24] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. “OpenFABMAP: An open source toolbox for appearance-based loop closure detection”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 4730–4735.

[25] C. G. Harris, M. Stephens, et al. “A combined corner and edge detector.” In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[26] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[27] M. Irani and P. Anandan. “About direct methods”. In: *International Workshop on Vision Algorithms*. Springer. 1999, pp. 267–277.

[28] 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.

[29] J. Kannala and S. S. Brandt. “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006), pp. 1335–1340.

[30] C. Kerl. “Odometry from rgb-d cameras for autonomous quadrocopters”. In: *Master’s Thesis, Technical University* (2012).

[31] C. Kerl, J. Sturm, and D. Cremers. “Dense visual SLAM for RGB-D cameras”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 2100–2106.

[32] C. Kerl, J. Sturm, and D. Cremers. “Robust odometry estimation for RGB-D cameras”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 3748–3754.

[33] B. Khomutenko, G. Garcia, and P. Martinet. “An enhanced unified camera model”. In: *IEEE Robotics and Automation Letters* 1.1 (2015), pp. 137–144.

[34] G. Klein and D. Murray. “Improving the agility of keyframe-based SLAM”. In: *European Conference on Computer Vision*. Springer. 2008, pp. 802–815.

[35] G. Klein and D. Murray. “Parallel tracking and mapping for small AR workspaces”. In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2007, pp. 1–10.

[36] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “g 2 o: A general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3607–3613.

[37] K. L. Lange, R. J. Little, and J. M. Taylor. “Robust statistical modeling using the t distribution”. In: *Journal of the American Statistical Association* 84.408 (1989), pp. 881–896.

[38] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[39] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzmán. “Accurate visual odometry from a rear parking camera”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2011, pp. 788–793.

[40] J. Loxam and T. Drummond. “Student-t mixture filter for robust, real-time visual tracking”. In: *European Conference on Computer Vision*. Springer. 2008, pp. 372–385.

[41] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. Springer Science & Business Media, 2012.

[42] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers. “Omnidirectional DSO: Direct sparse odometry with fisheye cameras”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3693–3700.

[43] C. Mei, S. Benhimane, E. Malis, and P. Rives. “Efficient homography-based tracking and 3-D reconstruction for single-viewpoint sensors”. In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1352–1364.

[44] C. Mei and P. Rives. “Single view point omnidirectional camera calibration from planar grids”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3945–3950.

[45] M. Meilland and A. I. Comport. “On unifying key-frame and voxel-based dense visual SLAM at large scales”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3677–3683.

[46] N. Molton, A. J. Davison, and I. D. Reid. “Locally Planar Patch Features for Real-Time Structure from Motion.” In: *Bmvc*. Citeseer. 2004, pp. 1–10.

[47] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[48] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2320–2327.

[49] M. Pizzoli, C. Forster, and D. Scaramuzza. “REMODE: Probabilistic, monocular dense reconstruction in real time”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2609–2616.

[50] A. Pretto, E. Menegatti, and E. Pagello. “Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3289–3296.

[51] W. Rossmann. *Lie groups: an introduction through linear groups*. Vol. 5. OUP Oxford, 2002.

[52] T. Schöps, J. Engel, and D. Cremers. “Semi-dense visual odometry for AR on a smartphone”. In: *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE. 2014, pp. 145–150.

[53] G. Silveira, E. Malis, and P. Rives. “An efficient direct approach to visual SLAM”. In: *IEEE transactions on robotics* 24.5 (2008), pp. 969–979.

[54] H. Strasdat. “Local accuracy and global consistency for efficient visual SLAM”. PhD thesis. Department of Computing, Imperial College London, 2012.

[55] H. Strasdat, J. Montiel, and A. J. Davison. “Scale drift-aware large scale monocular SLAM”. In: *Robotics: Science and Systems VI* 2.3 (2010), p. 7.

[56] J. Stühmer, S. Gumhold, and D. Cremers. “Real-time dense geometry from a handheld camera”. In: *Joint Pattern Recognition Symposium*. Springer. 2010, pp. 11–20.

[57] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 573–580.

[58] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[59] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. “Bundle adjustment—a modern synthesis”. In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[60] T. Tykkälä, C. Audras, and A. I. Comport. “Direct iterative closest point for real-time visual odometry”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Citeseer. 2011, pp. 2050–2056.

[61] V. Usenko, N. Demmel, and D. Cremers. “The Double Sphere Camera Model”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 552–560.

[62] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers. “Visual-Inertial Mapping with Non-Linear Factor Recovery”. In: *arXiv preprint arXiv:1904.06504* (2019).

[63] Z. Zhang. “Parameter estimation techniques: A tutorial with application to conic fitting”. In: *Image and vision Computing* 15.1 (1997), pp. 59–76.

[64] J. Zubizarreta, I. Aguinaga, and J. Montiel. “Direct Sparse Mapping”. In: *arXiv preprint arXiv:1904.06577* (2019).