RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Informatik 8 (Computer Vision)
Fakultät für Mathematik, Informatik und Naturwissenschaften
Prof. Dr. Bastian Leibe

Bachelor Thesis

Global Keyfame-Based Optimization for a Direct Sparse Visual Odometry Approach

vorgelegt von

Nikolay Paleshnikov
Matrikelnummer: 331623
2017-08-07

Erstgutachter:	Prof. Dr. Bastian Leibe
Zweitgutachter:	Prof. Dr. Leif Kobbelt
Advisor:	Dr. Jörg Stückler

Abstract

The recently developed direct sparse odometry framework DSO stands apart among all recent research on monocular visual odometry with its robustness and precision. It accumulates, however, considerable motion drift over the course of loops in the trajectory because it only employs local window optimization. We have recognized that a loop detection mechanism providing constraints for global keyframe pose optimization could reduce this motion drift and thus make the trajectory estimate globally consistent. We have therefore developed a global optimization layer for DSO, which initializes the variables of a factor graph with the DSO pose estimates for all keyframes, connects them whenever possible with photometric factors for further optimizing or between pose factors for maintaining their relative poses and enforces between pose constraints upon loop detections for eliminating the drift accumulated over the course of loops in the trajectory. We have hence defined our custom photometric factors for the optimization of a direct global energy formulation inspired by the photometric local optimization window energy defined in DSO and implemented its global optimization by means of the incremental smoothing and mapping algorithm iSAM2 from the GTSAM factor graph optimization library. We have also developed a loop detection mechanism building upon existing approaches for image matching based on ORB feature descriptors stored in a DBoW2 keyframe database and performing a geometric consistency check to filter out pairs of keyframes, for which a between pose constraint imposing a reliable rigid body transformation estimate is to be added to the factor graph. An evaluation of our method against DSO and the prominent simultaneous localization and mapping system ORB-SLAM shows its superiority in regard to the absolute trajectory and the relative pose error of its trajectory estimates for a monocular visual odometry dataset.

Acknowledgements

I would like to thank Prof. Dr. Bastian Leibe for sparking my interest in the field of Computer Vision, as well as my advisor Dr. Jörg Stückler for introducing me into his research on visual odometry and SLAM. I am also grateful to my family and friends, who always stood by me and brought me joy even in times of trouble. I am equally indebted to the German National Merit Foundation (die Studienstiftung des deutschen Volkes) for the financial and ideational support of my studies.

Eidesstattliche Versicherung

Nikolay Paleshnikov	331623
Name	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Bachelorarbeit mit dem Titel

Global Keyfame-Based Optimization for a Direct Sparse Visual Odometry Approach

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 2017-08-07	
Ort, Datum	Unterschrift

Belehrung:

§ **156 StGB: Falsche Versicherung an Eides Statt**
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ **161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt**

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten dementsprechend.

Die vorstehende Belehrung habe ich zur Kentnis genommen:

Aachen, 2017-08-07	
Ort, Datum	Unterschrift

Contents

1 Introduction

1.1 Monocular Visual Odometry and SLAM

Monocular visual odometry is a building block for many applications such as mobile robotics or augmented reality. In the absence of any knowledge about the three-dimensional environment, the initial position of a robot is used as a reference point in order to continually determine its current position. The egomotion of a single camera placed on the robot is estimated as three-dimensional transformations between successive timestamps, the concatenation of which approximates the three-dimensional transformation between the initial and the current position of the robot.

A more robust solution of the visual odometry problem can be obtained if we take multiple camera views into account, all of which share a substantial amount of points. This forms an optimization window of active keyframes at those camera views, changing as we move through the unknown environment. Nevertheless, the inaccuracies in each three-dimensional transformation estimation add up over time to produce a substantial drift between real and approximated robot position.

An useful extension to a visual odometry framework is to globally optimize these three-dimensional transformations, using additional cues such as detected loops in the trajectory. If the global optimization takes place on the fly, as well as a map of the environment modelled as a three-dimensional point cloud is continually updated, the resulting approach is called Simultaneous Localization and Mapping (SLAM).

1.2 Summary of Contributions

We have extended the monocular visual odometry framework DSO with a global keyframe-based optimization layer. To this end, we have employed the factor graph optimization library GTSAM and the optimizer iSAM2 in particular. We have created a novel photometric factor with an error function inspired

by the photometric local optimization window energy formulation of DSO. It is designed to capture the essence of the local window optimization performed by DSO and ensure that the global optimization layer remains consistent with it. We have also developed a loop detection mechanism to provide additional hints for the global optimization.

1.3 Thesis Structure

In Chapter 2, we outline some leading recent frameworks attempting to solve the localization problem formulated in the introduction. We first present two full SLAM systems in Section 2.1, emphasizing their shortcomings in comparison with the pure monocular visual odometry framework DSO, whose functional principle is described in Section 2.2 . Thereby we do not only consider the different design choices underlying these frameworks, but also take into account the quantitative comparison data presented in the respective publications. We end Chapter 2 with Section 2.3 and a brief description of the smoothing and mapping library GTSAM, which we employ for global optimization. In order to properly expound how the method we have devised works, we first summarize the used notation and the underlying theory in Chapter 3. There, we also explain the theory behind DSO and GTSAM in more detail. The pivotal Chapter 4 comes next. It is devoted to our approach to global keyframe-based optimization of the estimated keyframe camera poses for a given video sequence processed by DSO. In the following Chapter 5, we provide an evaluation of our loop detection routine and the global optimization. We present a quantitative comparison of our approach with the original DSO algorithm and ORB-SLAM based on the TUM monocular visual odometry dataset as well as some qualitative results. In the last Chapter 6, we summarize the advantages of our approach and propose directions for further research.

2 Related Work

2.1 SLAM Frameworks

Some excellent SLAM frameworks have been developed in recent years. The most notable difference is whether they employ a direct or an indirect, i.e. feature-based energy formulation. It results in two distinct algorithm step sequences as shown in Figure 2.1. While the feature-based algorithms extract features at each keyframe and use them to minimize a geometric reprojection error, direct ones define a photometric error based on actual sensor values. Instead of using interest points comprising a feature vector, direct frameworks evenly sample points with an intensity gradient above a given threshold, which enables them to produce a much denser map of the three-dimensional environment. Each of these two energy formulations with their respective keypoint extraction mechanisms underlies one of the monocular SLAM systems reviewed in the rest of this section.

2.1.1 LSD-SLAM

The Large-Scale Direct monocular SLAM (LSD-SLAM) algorithm proposed in [ESC14] uses a direct energy formulation and thus minimizes photometric errors. This is achieved by means of coarse-to-fine Gauss-Newton optimization of the photometric residuals arousing from sample point projections between adjacent keyframes. The three main algorithm blocks are shown in Figure 2.2. Tracking and depth map estimation are performed alternately in a main thread, while a separate map optimization thread runs in the background.

Tracking is performed by direct image alignment based on sample point correspondence search along epipolar lines. Thus the camera pose of a new keyframe is estimated and its depth map initialized. Subsequent frames from camera views close enough to the one of the last keyframe are used to further refine the depth map of the new keyframe. Once the difference between the current camera view and the one of the last keyframe becomes too large, a new keyframe is created. A sample point projection between the new frame and the current keyframe as

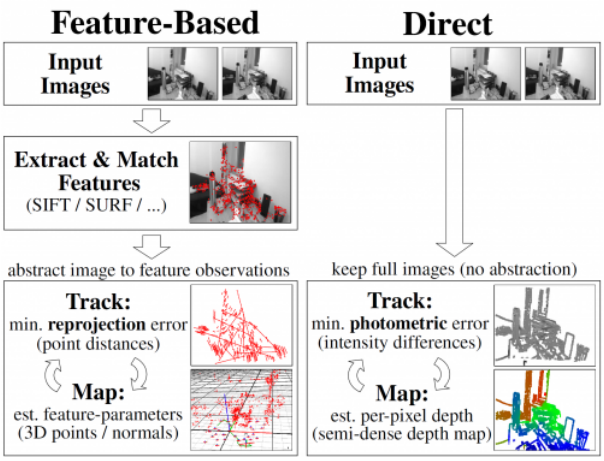


Figure 2.1: **A comparison of feature-based and direct SLAM formulations.** Direct SLAM methods skip the feature extraction and matching step and thus operate further on the input images themselves, while indirect systems ones only keep the extracted features for the subsequent tracking and mapping steps. Taken from [ESC14].

well as an image of photometric differences at all projected points is to be seen in Figure 2.3. This image forms the basis for direct $SE(3)$ image alignment performed by the tracking routine and for the $Sim(3)$ image alignment employed in the map optimization routine upon loop closure.

The estimated $SE(3)$ keyframe-to-keyframe transformations are enriched by $Sim(3)$ pose constraints derived from detected loops. Loop closing candidates among existing keyframes are proposed by an appearance-based matching algorithm and tested by means of a reciprocal tracking check as explained in section 3.5 of [ESC14]. Pose graph optimization over the estimated keyframe-to-keyframe three-dimensional transformations through both direct image alignment and loop closing is then performed in a dedicated thread in the background.

The inherent scale ambiguity of monocular visual odometry may result in a scale drift while processing a given video sequence. To counteract this, LSD-SLAM uses three-dimensional similarity transformations for the global optimization.

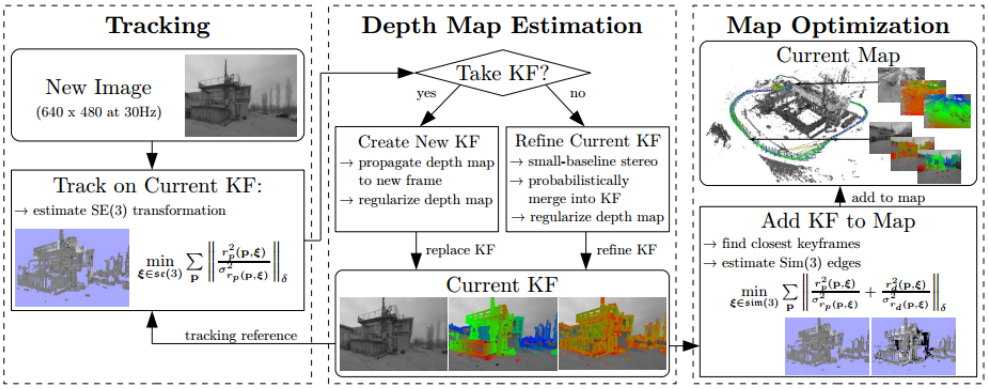


Figure 2.2: **An overview of LSD-SLAM.** The tracking routine delivers an $SE(3)$ transformation estimate between the new frame and the current keyframe, on the basis of which the depth map estimation routine decides whether the new frame has to become a new keyframe or is merely used to improve the depth map of the current keyframe. In case a new keyframe is created, for each loop closure candidate the map optimization routine estimates a $Sim(3)$ transformation. If it withstands a geometric consistency check, it is used as a constraint in the global pose graph optimization. Taken from [ESC14].

2.1.2 ORB-SLAM2

The revised ORB-SLAM2 algorithm presented in [MT16] as well as its predecessor ORB-SLAM [MAMT15] can also operate in a monocular setting. Applying an indirect approach in contrast to LSD-SLAM, it minimizes geometric errors. FAST corners [RD06] are extracted as interest points in each keyframe. Their scale- and rotation-invariant ORB feature descriptors [RRKB11] are subsequently computed and then used as a bag-of-words representation of the keyframe, which is inserted into a DBoW2 keyframe database [GLT12].

The algorithm simultaneously runs in three threads, each of them dedicated to respectively tracking, mapping and loop closing as visible in Figure 2.4. The tracking thread estimates the camera egomotion. It initializes the camera pose of a new frame by feature matching to the previous keyframe and then performs motion-only bundle adjustment, i.e. linear keyframe pose optimization assuming fixed three-dimensional point positions.

The mapping thread performs local bundle adjustment among keyframes connected to the current one in a covisibility graph so that they share a significant amount of interest points. New correspondences found among unmatched ORB descriptors of the current keyframe and the ORB descriptors of its neighbours in the covisibility graph are used to triangulate new points. Keyframes with a high percentage of interest points seen in at least three other keyframes at the same or finer scale are purged from the covisibility graph, wherefore the local bundle

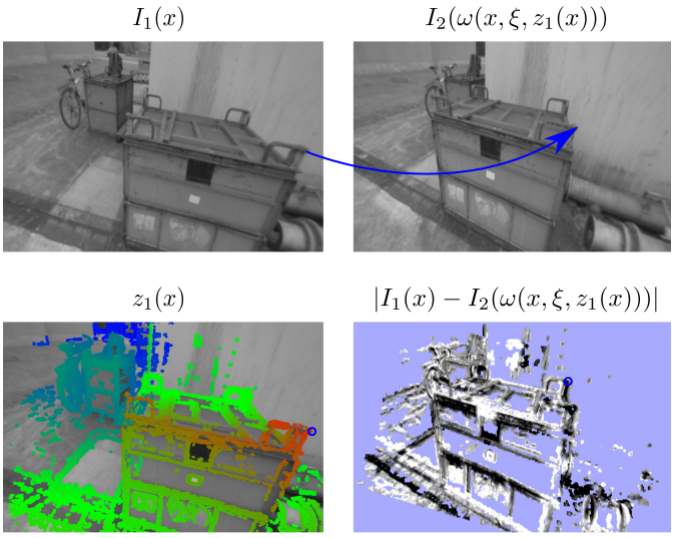


Figure 2.3: **Direct $SE(3)$ and $Sim(3)$ keyframe alignment.** In the top row, the projection of a single point between two given keyframes can be seen. The blue arrow connects the point position in the first image with its warped position in the second image. The point depth needed for the warping function can be inferred from the depth map of the first image shown on the left-hand side of the bottom row . On its right, we see an image of photometric differences between the two keyframes, at which the images in the bottom row have been taken. The value represented at the spot marked with a blue circle corresponds to the the grayscale value difference between the point in the first and the warped point in the second image. Adapted from [ESC14]

adjustment step is accelerated and a lifelong operation in the same environment is rendered feasible, as the number of keyframes only grows if the algorithm is presented with new visual content.

The loop closing thread ranks loop closing candidates according to the similarity of their bag-of-words vector to the vector of the current keyframe. If a three-dimensional similarity transformation with enough inliers among the interest points of the keyframe is found using RANSAC [FB81], a loop in the robot motion is detected. A pose graph optimization of a relevant subset of the covisibility graph is performed so as to achieve global consistency. Thereby both ends of the loop are aligned together to compensate for accumulated drift, newly triangulated point positions are corrected and duplicated points are fused as shown in Figure 2.5.

Like LSD-SLAM, ORB-SLAM2 operates on three-dimensional similarity transformations. It outperforms LSD-SLAM with respect to keyframe localization accuracy as stated in section VIII-B of [MAMT15], but its tracking alone has a

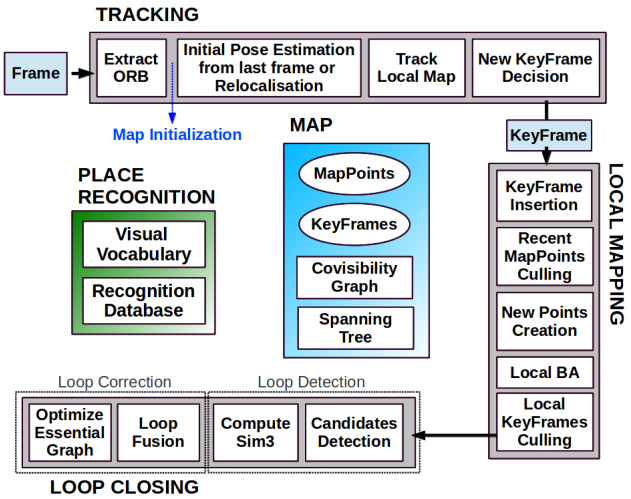
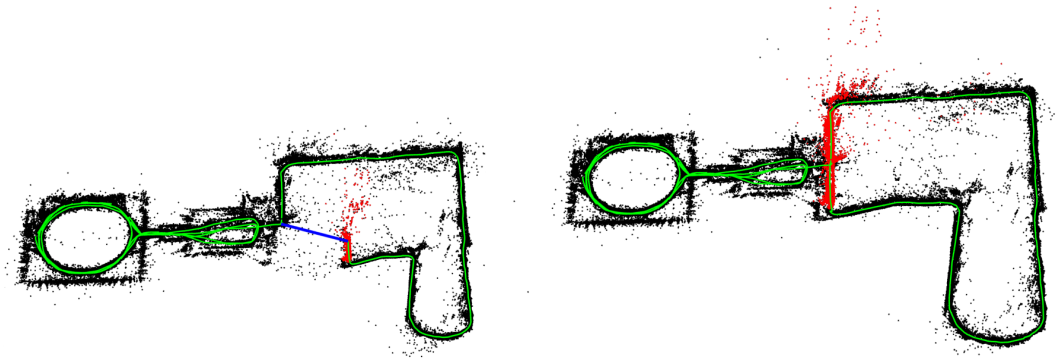


Figure 2.4: **An overview of ORB-SLAM.** The main steps of three main threads, each of which performs respectively tracking, mapping and loop closing, as well as their interactions with each other are shown alongside the place recognition and mapping modules. Taken from [RRKB11].



(a) Loop detection (b) Loop closing

Figure 2.5: **Loop detection and closing in ORB-SLAM.** On the left, the trajectory is shown in green just after a loop has been detected. On the right, the updated trajectory can be seen after the loop has been closed. Note that not only do keyframes near the loop change their positions, but also newly triangulated points positions (shown in red) are corrected and duplicated points are fused. Taken from [RRKB11].

bigger alignment error over the whole trajectory as defined in [EUC16] than DSO (section 4.1 of [EKC17]).

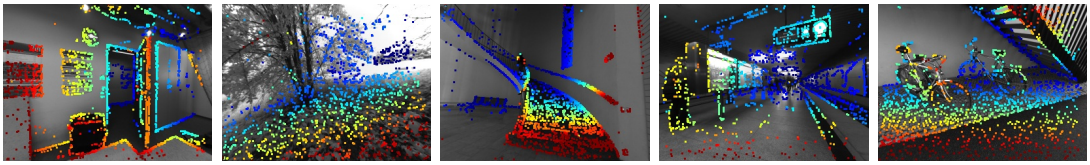


Figure 2.6: **Depth maps estimated by DSO.** Taken from [EKC17].

2.2 DSO

The monocular visual odometry framework we are building upon is presented next. It has been proposed in [EKC17] on the basis of a Direct Sparse visual Odometry (DSO) formulation. It does not rely on keypoint detectors or descriptors, but instead evenly samples pixels from all regions with sufficiently large intensity gradient such as edges or smooth intensity variations on walls. As a preliminary step, DSO photometrically corrects each frame as described in Section 3.3.

As a consequence of a changing field of view, occlusions, disocclusions or significant changes in exposure time, a new keyframe is taken and old ones are marginalized. The geometric parameters including camera poses and sample point depths as well as the photometric parameters used for affine brightness transfer and the camera intrinsic parameters are then jointly optimized. This is done by means of coarse-to-fine Gauss-Newton optimization of the photometric errors of projecting each sample point in one keyframe from the optimization window into each of the others as thoroughly explained in Sections 3.4 and 3.6.

As the optimization window moves, sample points not having been observed in the last two keyframes and keyframes with too few active points become marginalized alternately using the Schur complement as expounded in section 2.3 of [EKC17]. In keeping only a fixed number of active keyframes and a fixed number of sample points equally spread across the active keyframes and over space, DSO maintains the sparsity pattern of the optimization.

Since DSO only manages local optimization windows, it does not attempt to fix the scale drift accumulated while estimating the trajectory and thus uses rigid body transformations. It deals reliably with a multitude of environments featuring highly fluctuating exposure times and big scale changes over the course of the trajectory. Some sample depth maps estimated by DSO at frames taken from the TUM monocular visual odometry dataset [EUC16] are shown in Figure 2.6. The semi-dense point-cloud of a three-dimensional scene taken from the same dataset alongside two depthmaps at keyframes observing the scene is shown in 2.7.

Non-linear Gauss-Newton optimization instead of camera pose linearization or point depth triangulation enables DSO to outperform indirect approaches and ORB-SLAM2 [MAMT15] in particular in the camera egomotion estimation accuracy (section 4.1 of [EKC17]) if provided with photometrically well-

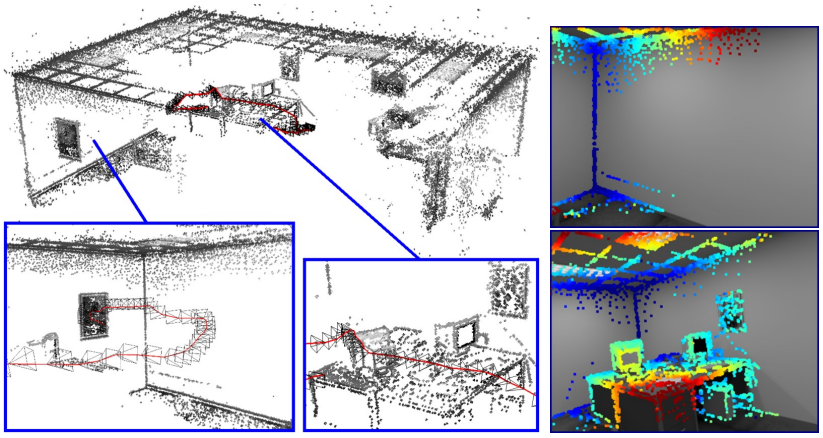


Figure 2.7: **Semi-dense point cloud estimation delivered by DSO.** On the left, two camera views at keyframes created by DSO are zoomed into. On the right, their corresponding depth maps are shown. Adapted from [EKC17].

calibrated data such as the TUM monocular visual odometry benchmark presented in [EUC16].

2.3 GTSAM

The Georgia Tech Smoothing and Mapping library (GTSAM) is a C++ toolbox developed with the aim of solving many common Smoothing and Mapping (SAM) problems in the field of computer vision. It implements the square root SAM technique described in [DK06], which is particularly well suited for solving the SLAM problem presented in Section 2.1, as well as for global optimization in general. We employ the factor graph formulation for our global optimization problem and use the incremental Smoothing and Mapping 2 algorithm (iSAM2) presented in [KJR+11] to solve it. iSAM2 implements a fully incremental sparse matrix factorization employing the Bayes tree as a data structure. It brings about major efficiency benefits in comparison to a simple batch optimization by selectively relinearizing factor graph variables whenever necessary.

3 Preliminary Theory and Notation Summary

This chapter summarizes the theory underlying the rest of the thesis. In case the reader needs further details, we point out section 2 of [EKC17] as a reference for Sections 3.2 – 3.4, 3.5.2 and 3.6. The GTSAM documentation provided with the library offers an in-depth treatment of the concepts presented in the remaining sections of this chapter, as well as in Section 3.6.

3.1 Three-Dimensional Representation of a Point and a Transformation

Unless stated otherwise, we use augmented three-dimensional point coordinates:

$$p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4$$

An $SE(3)$ transformation consisting of a rotation $R \in \mathbb{R}^{3\times 3}$ and a translation $t \in \mathbb{R}^3$ is represented by the matrix:

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

which defines the linear map $M_T : \mathbb{R}^4 \to \mathbb{R}^4$, $p \mapsto T\,p$.

3.2 Point Projection

We write points in the image plane of a given camera in bold:

$$\mathbf{p} = \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^2$$

Point projection is denoted by:

$$\Pi : \mathbb{R}^4 \to \mathbb{R}^2 \, , \, \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} x/z \\ y/z \end{pmatrix} \tag{3.1}$$

Its reverse is defined as:

$$\Pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^4 \, , \, (\begin{pmatrix} u \\ v \end{pmatrix}, d_p) \mapsto \begin{pmatrix} u/d_p \\ v/d_p \\ 1/d_p \\ 1 \end{pmatrix}$$

whereby $d_p = z^{-1}$ stands for the inverse depth of an image point p. For an account of the benefits of using an inverse depth parametrization, see [CDM08].

The camera intrinsic parameters comprise the local coordinates of the principal point (c_x, c_y), as well as the focal distance along the x and y axes: f_x and f_y. They form the camera matrix:

$$K = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Let $T_i \in R^{4\times4}$ denote the transformation matrix of camera i, transforming from camera point coordinates to world point coordinates. It also represents the $SE(3)$ pose of the camera in world coordinates. The matrix $T_i^j \in R^{4\times4}$ of the transformation from camera i to camera j can then be computed as:

$$T_i^j = T_j^{-1} \, T_i \tag{3.2}$$

The projection $\mathbf{p}'$ in frame j of a point $\mathbf{p}$ in frame i with inverse depth d_p can be obtained using the following equation:

$$\mathbf{p}' = \Pi \left(K \, T_i^j \, K^{-1} \, \Pi^{-1} \left(\mathbf{p}, d_p \right) \right) \tag{3.3}$$

3.3 Photometric Correction of a Video Frame

Let Ω be a video frame represented as a two-dimensional field of grayscale pixel intensities. Each pixel intensity is the result of a non-linear response function $G : \mathbb{R} \to [0, 255]$ so that the image $I_i : \Omega \to [0, 255]$ taken at camera frame i can be interpreted as a function of the irradiance $B_i : \Omega \to \mathbb{R}$ and the lens attenuation or vignetting $V : \Omega \to [0, 1]$ at each pixel $\mathbf{x}$ in the frame, as well as the frame exposure time t_i:

$$I_i(\mathbf{x}) = G(t_i \, V(x) \, B_i(\mathbf{x})) \tag{3.4}$$

The photometrically corrected frame $I'_i = t_i B_i(\mathbf{x})$ is therefore obtained by computing:

$$I'_i = \frac{G^{-1}(I_i(\mathbf{x}))}{V(\mathbf{x})} \tag{3.5}$$

In the rest of the thesis, we simply write I_i to denote the photometrically corrected image at frame i.

3.4 Photometric Error of a Local Optimization Window as Defined in DSO

In case exposure times t_i are unknown, an affine brightness transfer function $abt : I'_i \mapsto e^{-a_i}(I'_i - b_i)$ is estimated by finding optimal parameters a_i and b_i for each frame. The photometric error of a single point projection from keyframe i into keyframe j is then defined as follows:

$$E_{\mathbf{p}ij} = \sum_{\mathbf{q} \in N_\mathbf{p}} \left\| (I_j[\mathbf{q}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{q}] - b_i) \right\|_\gamma \tag{3.6}$$

where $\mathbf{p}$ is the point to be projected, $N_\mathbf{p}$ – a residual pattern of neighbouring points of $\mathbf{p}$ as shown in Figure 3.1 and $\mathbf{q}'$ is defined as in Equation 3.3. t_i, t_j, a_i, a_j, b_i and b_j refer to the variables already seen in Section 3.3. $\|\|_\gamma$ stands for the Huber norm L_δ defined for a given threshold δ as follows:

$$L_\delta(r) = \begin{cases} \frac{1}{2} r^2 & \text{if } |r| < \delta \\ \delta \, |r| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{3.7}$$

The photometric error of a whole optimization window w is given by:

$$E_w = \sum_{i \in F} \sum_{\mathbf{p} \in P_i} \sum_{j \in obs(\mathbf{p}, i)} E_{\mathbf{p}ij} \tag{3.8}$$

where F is the set of active keyframes, i.e. keyframes in the local optimization window, P_i the set of points in keyframe i and $obs(\mathbf{p}, i) \subseteq F \setminus \{i\}$ the set of active

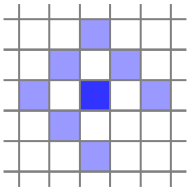


Figure 3.1: **The residual pattern** of neighbouring points used in DSO. Taken from [EKC17].

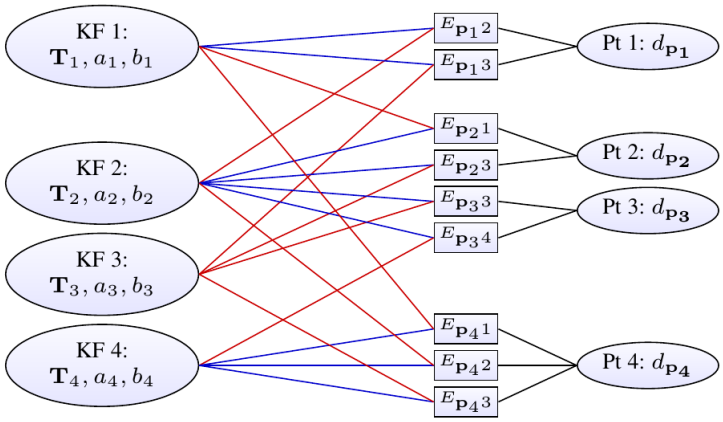


Figure 3.2: **A factor graph for the photometric error of a whole optimization window.** Each factor, i.e. each photometric error of a single point projection, depends on its reference frame (blue edge), target frame (red edge), the point's inverse depth (black edge) and on the camera intrinsic parameters, which are abstracted away in this illustration. Taken from [EKC17].

keyframes other than i, into which each point in the residual pattern of $\mathbf{p}$ can be projected. $E_{\mathbf{p}ij}$ is defined as in Equation 3.6. A sample factor graph for a small optimization window with four keyframes containing four points in total is shown in Figure 3.2.

3.5 Parameter Space Used for the Gauss-Newton Optimization

3.5.1 Lie Group, Lie Algebra and Twist Coordinates

Lie groups provide us with a useful representation for the camera pose optimization. A Lie group G is both a group and a smooth manifold. Its corresponding Lie algebra $\mathfrak{g}$ is related to the Lie group by means of a bijective exponential map:

$$exp : \mathfrak{g} \to G \ , \ \xi \mapsto g$$

which maps each element of the tangent space $\xi \in \mathfrak{g}$ onto its corresponding Lie group element $g \in G$.

The $SE(3)$ transformation matrices from camera point coordinates to world point coordinates form a closed subgroup of the general linear group $GL(6)$ and can therefore be handled as a matrix Lie group $G_{SE(3)}$. The tangent space of its increments at identity forms the Lie algebra $\mathfrak{g}_{SE(3)}$, which also has six degrees of freedom. We can therefore define the bijective 'hat operator':

$$\hat{}: \mathbb{R}^6 \to \mathfrak{g} \ , \ \begin{pmatrix} \omega \\ \nu \end{pmatrix} \mapsto \begin{pmatrix} [\omega]_\times & \nu \\ 0 \ \ 0 \ \ 0 & 0 \end{pmatrix}$$

with

$$\left[\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \right]_\times$$

defined as the skew-symmetric matrix

$$\begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{3.9}$$

The vector

$$\begin{pmatrix} \omega \\ \nu \end{pmatrix} \in \mathbb{R}^6$$

corresponding to each Lie algebra element

$$\widehat{\begin{pmatrix} \omega \\ \nu \end{pmatrix}} \in \mathfrak{g}$$

is called its *twist vector*. It stores the *twist coordinates* ω representing the rotational part and ν representing the translational part of an $SE(3)$ transformation.

In the rest of this section, we are solely concerned with matrix Lie groups and assume they have n degrees of freedom.

3.5.2 Parameter Space in DSO

As indicated in Section 2.2, DSO jointly optimizes over the geometric and the photometric parameters. Keyframe camera poses are handled as elements of the matrix Lie group $G_{SE(3)}$, whereas sample point depths, affine brightness transfer and camera intrinsic parameters are simply represented as real numbers. All optimized variables can then be denoted as $\zeta \in G_{SE(3)}^k \times \mathbb{R}^r$ with k indicating the

number of active keyframes and r – the number of sample points, affine brightness transfer and camera intrinsic parameters altogether.

3.5.3 Lie Group Generators

The basis elements of a Lie algebra are called its *generators*. Not only are all tangent vectors of the corresponding Lie group their linear combinations but they also serve as partial derivatives along each degree of freedom of the group. In the case of $\mathfrak{g}_{SE(3)}$, which has six degrees of freedom, we can therefore use the following six generators ordered according to the convention used in GTSAM (first three for rotation, next three for translation):

$$g_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad g_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad g_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$g_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad g_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad g_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

3.5.4 Adjoint Map

Since we do not only want to apply increments at the origin, i.e. at the neutral group element, which is the identity for matrix Lie groups, we are interested in a map between different tangent spaces. It saves us the computation of conjugate matrices for actions applicable at group elements other than identity and is known by the name of an *adjoint map*. Each group element $T \in G$ has its respective adjoint map $Ad_T : \mathfrak{g} \to \mathfrak{g}$, which receives an increment $\hat{\xi}$ defined at the origin and transforms it into one that can be applied directly at T so that:

$$T\ exp(\hat{\xi})\ T^{-1} = exp(Ad_T\ \hat{\xi})$$

For matrix Lie groups, we also have:

$$Ad_T\ \hat{\xi} = T\ \hat{\xi}\ T^{-1} \tag{3.10}$$

wherefore

$$T\ exp(\hat{\xi})\ T^{-1} = exp(T\ \hat{\xi}\ T^{-1}) \tag{3.11}$$

The adjoint map Ad_T of a transformation $T \in G_{SE(3)}$ with rotation $R \in \mathbb{R}^{3\times3}$ and translation $t \in \mathbb{R}^3$ is thus

$$
\begin{aligned}
Ad_T\,\hat{\xi} = T\,\hat{\xi}\,T^{-1} &= \begin{pmatrix} R & t \\ 0\ 0\ 0 & 1 \end{pmatrix} \begin{pmatrix} [\omega]_\times & \nu \\ 0\ 0\ 0 & 0 \end{pmatrix} \begin{pmatrix} R^T & -R^T\,t \\ 0\ 0\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} [R\,\omega]_\times & -[R\,\omega]_\times\,t + R\,\nu \\ 0\ 0\ 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} [R\,\omega]_\times & t \times R\,\omega + R\,\nu \\ 0\ 0\ 0 & 0 \end{pmatrix}
\end{aligned}
$$

with $[\omega]_\times$ defined in Equation 3.9.

Expressed as a linear map, this is equivalent to

$$
\begin{pmatrix} \omega \\ \nu \end{pmatrix} \mapsto \begin{pmatrix} R & 0 \\ [t]_\times R & R \end{pmatrix} \begin{pmatrix} \omega \\ \nu \end{pmatrix} \tag{3.12}
$$

with $[t]_\times$ defined similarly to $[\omega]_\times$.

3.5.5 Increments

In order to apply an increment at $a \in G$ with twist coordinates ξ, we use the right-multiplicative convention used in GTSAM contrary to the left-multiplicative formulation employed by DSO and define the infix increment operator of arity two:

$$
\oplus : G \times \mathbb{R}^n \to G\ ,\ (a, \xi) \mapsto a\ exp(\hat{\xi}) \tag{3.13}
$$

which has precedence over matrix multiplication.

Since DSO optimizes further parameters than camera poses, we have to extend the notion of an increment for its parameter space. An increment $x = ((\xi^i)_{i \in F}, w) \in (\mathbb{R}^6)^k \times \mathbb{R}^r$ can be handled as a tuple of the twist vector coordinates of the increments at each frame in the set of active keyframes F with cardinality $|F| = k$ and real numbers representing additive increments for the rest of the optimized parameters. Its application at a given variable set $\zeta_0 = ((g^i)_{i \in F}, v) \in G^k \times \mathbb{R}^r$ can then be modelled as the operator:

$$
\begin{aligned}
\boxplus : ((G^i_{SE(3)})_{i \in F} \times \mathbb{R}^r) \quad &\times \quad ((\mathbb{R}^6)^k \times \mathbb{R}^r) \to \quad ((G^i_{SE(3)})_{i \in F} \times \mathbb{R}^r), \\
(((g^i)_{i \in F}, v) \quad &, \quad ((\xi^i)_{i \in F}, w)) \mapsto \quad ((g^i\ exp(\hat{\xi}^i))_{i \in F}, v + w).
\end{aligned} \tag{3.14}
$$

The relationship between the Lie group $G_{SE(3)}$, its tangent space, i.e. the Lie algebra $\mathfrak{g}_{SE(3)}$, the current variable set ζ_0 and the updated variable set ζ after the application of the increment x is represented in Figure 3.3. The additional increment δ is dealt with in Section 3.6.2.

Because the marginalization of a residual fixes the tangent space of all the parameters involved in its computation, we let ζ_0 stand for its evaluation point and apply the accumulated increments x onto it to obtain the current state estimate $\zeta = \zeta_0 \boxplus x$

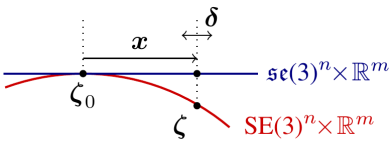


Figure 3.3: **Increment application at an SE(3) pose.** Taken from [EKC17].

3.5.6 Derivatives and Jacobian Matrix

As an approximation of a function $f : G \to \mathbb{R}^m$ in a neighbourhood around $a \in G$, we define the Jacobian matrix $f'(a) \in \mathbb{R}^{m \times n}$ of f at $a = (a^1, ..., a^n)$ such that:

$$\lim_{\xi \to 0} \frac{|f(a) + f'(a)\,\xi - f(a \oplus \xi))|}{|\xi|} = 0$$

The linear map $Df_a : \xi \mapsto f'(a)\,\xi$ is then called the derivative of f at a.
The partial derivative $D_j f^i(a)$ is defined as follows:

$$D_j f^i(a) = \lim_{h \to 0} \frac{f^i(a^1, ..., a^j + h, ..., a^n) - f^i(a)}{h} = 0$$

The Jacobian matrix $f'(a)$ can then be computed elementwise as the partial derivatives $D_j f^i(a)\ \forall i \in [1, ..., m]\ \forall j \in [1, ..., n]$ to obtain:

$$f'(a) = \begin{pmatrix} D_1 f^1(a) & ... & D_n f^1(a) \\ \vdots & \ddots & \vdots \\ D_1 f^m(a) & ... & D_n f^m(a) \end{pmatrix}$$

We will sometimes write F_a instead of $f'(a)$ to stress the fact that we are dealing with a matrix.

3.5.7 Pushforward

Let $\phi : G \to G\ ,\ a \mapsto b$ be a smooth map between elements of the Lie group G with tangent space $\mathfrak{g}$. The linear map $\Phi_* : \mathfrak{g} \to \mathfrak{g}\ ,\ \hat{\xi} \mapsto \phi_*(\hat{\xi})$ defines the *pushforward* of a tangent vector $\hat{\xi} \in \mathfrak{g}$ such that $a\ exp(\hat{\xi}) = b\ exp(\phi_*(\hat{\xi}))$. By means of it we can approximate ϕ in a local neighbourhood around a:

$$\phi(a \oplus \xi) \approx \phi(a)\ exp(\phi_*(\hat{\xi})) \tag{3.15}$$

Let us define the function composition operator:

$$\circ : (Y \to Z) \times (X \to Y) \to (X \to Z)\ ,\ (g, f) \mapsto h$$

We can regard the transformation f from camera i to camera j with matrix $T_i^j \in G$ defined in 3.2 as the composition $\psi^{-1} \circ \phi$ of the linear maps ψ with transformation matrix $T_j \in G$ and ϕ with transformation matrix $T_i \in G$.

The pushforward with respect to ψ can then be obtained from the equality:

$$T_i^j \oplus y \stackrel{!}{=} (T_j \oplus x)^{-1} T_i$$

in which we have set $\hat{y} = \psi_*(\hat{x})$. It can be rearranged as follows:

$$T_j^{-1} T_i \, exp(\hat{y}) = (T_j \, exp(\hat{x}))^{-1} T_i$$

$$T_j^{-1} T_i \, exp(\hat{y}) = exp(\hat{x})^{-1} T_j^{-1} T_i$$

$$T_j^{-1} T_i \, exp(\hat{y}) = -exp(\hat{x}) \, T_j^{-1} T_i$$

$$exp(\hat{y}) = -T_i^{-1} T_j \, exp(\hat{x}) \, (T_i^{-1} T_j)^{-1}$$

Taking into account Equation 3.11, we can transform it into:

$$exp(\hat{y}) = -exp(T_i^{-1} T_j \, \hat{x} \, (T_i^{-1} T_j)^{-1})$$

By applying Equation 3.10 and taking into account that $f^{-1} = \phi^{-1} \circ \psi$, we obtain:

$$exp(\hat{y}) = -exp(Ad_{f^{-1}} \hat{x})$$

which is equivalent to:

$$\hat{y} = -Ad_{f^{-1}} \hat{x}$$

so that the pushforward coincides with the negated adjoint map at f^{-1}.

The pushforward with respect to ϕ can similarly be obtained from the equality

$$T_i^j \oplus y \stackrel{!}{=} T_j^{-1} T_i \oplus x$$

where we have set $\hat{y} = \phi_*(\hat{x})$. It yields $\hat{y} = \hat{x}$ so that the pushforward coincides with the identity mapping.

3.6 Gauss-Newton Optimization

3.6.1 Objective Function Linearization

The non-linear objective function of the local window Gauss-Newton optimization performed by DSO can be written as

$$h : (G^n)_{n \in F} \rightarrow \mathbb{R}^m \, , \; (g^n)_{n \in F} \mapsto R$$

with F denoting the set of local optimization window keyframes with cardinality $|F| = k$, R – the residual vector with cardinality $m = |R|$, comprising the

stacked summands of the single point projection errors $E_{\mathbf{p}ij}$ in the photometric error of a local optimization window E_w as defined in Equation 3.8, whereby we use the poses in the tuple $(g^n)_{n\in F}$ to compute each residual:

$$(I_j\ [\Pi\ (K\ (g^j)^{-1}\ g^i\ K^{-1}\ \Pi^{-1}\ (\mathbf{q}, d_q))] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\mathbf{q}] - b_i)$$

For the objective function minimization, an $x^* \in (G^n)_{n\in F}$ must be found such that:

$$x^* = arg\min_x \|h(x)\|_2^2$$

The objective function can be linearized around $a \in (G^n)_{n\in F}$ by means of the Jacobian matrix $H_a \in \mathbb{R}^{m\times k}$ of h at a such that:

$$h(a \oplus \xi) \approx h(a) + H_a\ \xi$$

Given an initialization $a \in (G^n)_{n\in F}$, we can therefore minimize the objective function with respect to small increments in the local neighbourhood of a:

$$\xi^* = arg\min_\xi \|h(a) + H_a\ \xi\|_2^2$$

Setting its first derivative to zero yields the normal equations:

$$H_a^T\ H_a\ \xi = -H_a^T\ h(a) \tag{3.16}$$

Note that in DSO, a diagonal matrix $W \in \mathbb{R}^{m\times m}$ is used to shift the optimization towards certain directions by amending the normal equations:

$$H_a^T\ W\ H_a\ \xi = -H_a^T\ W\ h(a) \tag{3.17}$$

The Levenberg-Marquardt formulation enhances the stability of the optimization in case of an initialization far from the optimum by introducing an adaptive damping parameter λ and further revising the normal equations to

$$(H_a^T\ W\ H_a\ +\ \lambda\ I)\ \xi = -H_a^T\ W\ h(a) \tag{3.18}$$

with $I \in \mathbb{R}^{k\times k}$ designating an identity matrix. By choosing a big λ in the beginning, it emulates the steepest descent method when the current solution is far from a local minimum to guarantee convergence and lowers λ in successive steps to accelerate convergence close to the local minimum.

3.6.2 Gauss-Newton Step

Each update step refers to a single local window optimization. It is performed upon addition of a new keyframe to the local optimization window. For the preservation of the sparsity of the optimization, this addition coincides with the

marginalization of keyframes by means of the Schur complement as explained in section 2.3 of [EKC17] such that their tangent spaces get fixed and all residuals based on points using one of them as a reference frame get dropped.

During Gauss-Newton optimization, residuals are always evaluated at the current estimate $\zeta_0 \boxplus x$ defined in Section 3.5.5. Each element h_{ij} of the Jacobian H_a is computed as the partial derivative $D_j f^i(\zeta_0 \boxplus x)$. The current increment δ is then the solution of the weighted normal equations from 3.17. It is added to the accumulated increment to obtain $x^{new} \leftarrow x + \delta$.

At the end of the update step, all variables that have not been marginalized yet are updated according to x so that $\zeta_0^{new} \leftarrow \zeta_0 \boxplus x$ and x is set to zero.

3.7 Epipolar Geometry and the Fundamental Matrix

In this section, we summarize the central concepts of epipolar geometry and provide an intuitive derivation of the fundamental matrix. We denote points as vectors $x \in \mathbb{R}^3$, represented by their world coordinates, and define an abridged version of the camera matrix from Section 3.2:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3}$$

For a more detailed treatment, the reader is kindly referred to [HZ04].

3.7.1 Epipolar Geometry

The *base line* connects both camera centers c and c' as shown in Figure 3.4. Its intersection with the image plane of a camera defines the camera *epipole* – e for the camera with center c and e' for the camera with center c'. It can also be seen as the projection of the center of one camera into the image plane of the other camera. The point X in three-dimensional space, its projections x and x' into the image planes of both cameras and both camera centers c and c' lie on the same plane π, which is known by the name of *epipolar plane*. On it also lies the *epipolar line* l' of x, which can be seen as the projection of the ray originating at the camera center c and passing through x into the image plane of the camera with center c'.

3.7.2 The Fundamental Matrix

The fundamental matrix F defines the linear map $M_F : x \mapsto l'$ for a point x and the epipolar line l' through its projection x' as shown in Figure 3.5. Since x' lies on l', we also have ${x'}^T\, l' = 0$ and thus ${x'}^T\, F\, x = 0$. It can be shown that F is the unique matrix in $\mathbb{R}^{3\times 3}$ with rank 2 that satisfies this equation for all pairs of a point x and its projection x'.

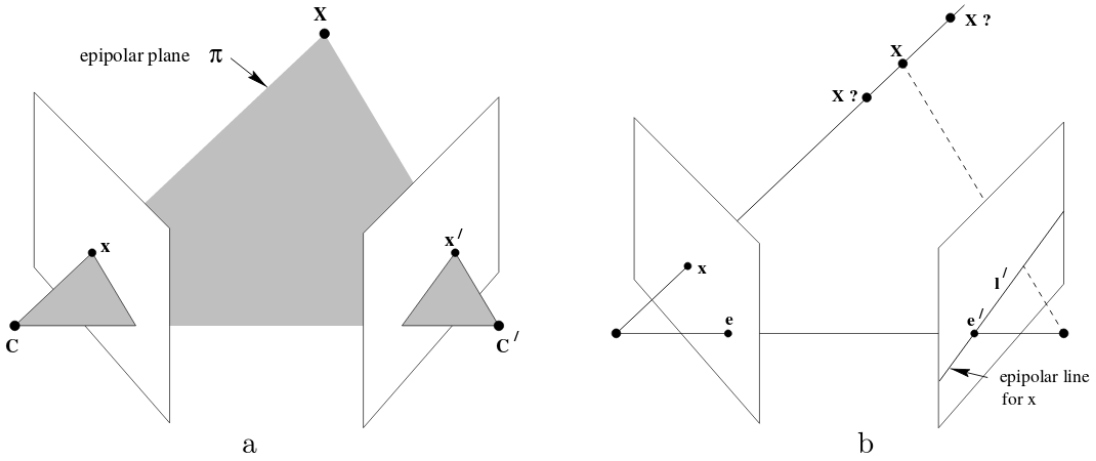


Figure 3.4: **Point projection, the epipolar plane and an epipolar line.** The epipolar plane is defined through the point X in space and both camera centers c and c'. The epipolar line l' of a given point x in the image plane of the reference camera with center c can be obtained as the intersection of the epipolar plane π with the image plane of the target camera with center c', into which the point is projected. Taken from [HZ04].

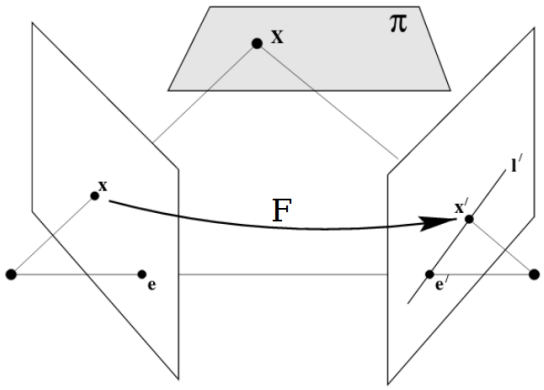


Figure 3.5: **The fundamental matrix** F as a linear map from a point x in the image plane of one camera to the epipolar line l' through the projected point x' in the image plane of the other camera. Adapted from [HZ04].

By denoting the rotation between both cameras as R and the respective translation – as t, as well as the camera matrix corresponding to the camera with center c – as K and the one corresponding to the camera with center c' – as K', we can write the projection from point x to point x' algebraically:

$$K'^{-1}\, x' = R\, K^{-1}\, x + t$$

We can then transform this equation by taking the cross-product with t on both sides, which is distributive over addition:

$$t \times K'^{-1}\, x' = t \times R\, K^{-1}\, x + \underbrace{t \times t}_{=0}$$

Note that the cross-product of a vector with itself equals zero. We further transform the equation by a left multiplication on both sides with the point, represented in three-dimensional space in the local frame of the camera with center c', i.e. $(K'^{-1}\, x')^T$:

$$\underbrace{(K'^{-1}\, x')^T\, t \times K'^{-1}\, x'}_{=0} = (K'^{-1}\, x')^T\, t \times R\, K^{-1}\, x$$

Because the three vectors on the left lie on the same plane, the left-hand side of the equation equals zero and we obtain

$$x'^T\, K'^{-T}\, [t]_\times\, R\, K^{-1}\, x = 0$$

with $[t]_\times$ as in Equation 3.12. Since the fundamental matrix F is unique, it necessarily follows that $F = K'^{-T}\, [t]_\times\, R\, K^{-1}$.

Note that each epipolar line is the intersection between the epipolar plane and the image plane and the epipole lies at the intersection of the baseline, which itself lies in the epipolar plane, and the image plane. Therefore the epipole lies on all epipolar lines simultaneously and we have:

$$e'^T\, l' = 0\ \forall l' \Rightarrow e'^T\, F\, x = 0\ \forall x \Rightarrow e'^T\, F = 0$$

We can similarly show that e belongs to the left null space of F. The fundamental matrix is thus singular and has rank 2 because each of the columns of $[t]_\times$ is a linear combination of both the others.

3.7.3 Normalized Eight-Point Algorithm

We often need to estimate the $SE(3)$ transformation between two given images taken from different three-dimensional camera poses. This can be done by means of the fundamental matrix. Given eight point correspondences $(x_i, x_i')_{i\in[1,...,8]}$ found e.g. by descriptor matching with each x_i lying in the image plane of one camera and each x_i' in the image plane of the other, we apply transformations T and T' to normalize them so that their centroids lie at the origin of the coordinate system and the root mean square distance equals $\sqrt{2}$. We obtain a system of eight linear equations $(T'x_i')^T\ F\ Tx\ \forall i \in [1, ..., 8]$, each of which can be reordered so as to have the entries of the fundamental matrix

$$F = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \in \mathbb{R}^{3\times3}$$

as unknowns

$$(u_i'u_i, u_i'v_i, u_i', v_i'u_i, v_i'v_i, v_i', u_i, v_i, 1)\ f = 0$$

whereby

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = Tx_i$$

$$\begin{pmatrix} u_i' \\ v_i' \end{pmatrix} = T'x_i'$$

and $f \in \mathbb{R}^9$ comprises the entries of F in row-major order. This defines the homogeneous linear system $A\ f = 0$, whose solution is obtained by computing the Singular Value Decomposition (SVD) of $A = U\ D\ V^T$ and taking the smallest singular vector of A, i.e. the third column of V, divided by its last element for normalization:

$$f = \begin{pmatrix} \frac{v_{19}}{v_{99}} \\ \frac{v_{29}}{v_{99}} \\ \vdots \\ 1 \end{pmatrix}$$

We transform f back into the matrix $\mathcal{F} \in R^{3\times3}$. Since the fundamental matrix must have rank 2 as motivated in Section 3.7.2, we compute the SVD of $\mathcal{F}$

$$\mathcal{F} = \mathcal{U} \begin{pmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{pmatrix} \mathcal{V}^T$$

and set:

$$F = T'^{\,T}\,\mathcal{U}\begin{pmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathcal{V}^T\,T$$

Note that we set the last singular value d_{33} to zero so as to obtain the best least-squares approximation of $\mathcal{F}$ in the sense of the Frobenius norm and multiply by $T'^{\,T}$ from the left and T from the right as a denormalization step in order to obtain a fundamental matrix corresponding to the original point matches.

4 Our Approach

4.1 Motivation

A pure visual odometry approach has the advantage of performing only local window optimization, which enables it to seamlessly run under real-time settings. However, the errors in the local three-dimensional transformation estimates add up over time resulting in a considerable drift between ground truth and estimated robot position as visible in Figure 4.1. Our solution is to integrate a global optimization layer with loop closing into DSO. We describe how it works in the rest of this chapter.

4.2 Global Optimization

4.2.1 Global Energy Formulation

We define K to be a set of ordered keyframe pairs. For each of its members (i, j) there exists at least one local optimization window created by DSO as it processes the video sequence, which contains both keyframes i and j. It necessarily follows that the pair (j, i) is also a member of the set.

Based on the local optimization window energy definition from Equation 3.8, we define the global energy as follows:

$$E_{global} = \sum_{(i,j)\in K} \sum_{\mathbf{p}\in pr(i,j)} \sum_{\mathbf{q}\in N_{\mathbf{p}}} w_\delta \left((I_j[\mathbf{q}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{q}] - b_i) \right) \tag{4.1}$$

where i is the reference frame, j – the target frame and $pr(i, j) \subseteq P_i$ denotes the subset of marginalized points of frame i, each of whose residual pattern points can be projected into the image plane of frame j. $N_{\mathbf{p}}$ is a residual pattern of neighbouring points of $\mathbf{p}$ as shown in Figure 3.1 and $\mathbf{q}'$ is defined as in Equation 3.3. t_i and t_j stand for exposure times, whereas a_i, a_j, b_i and b_j are the affine

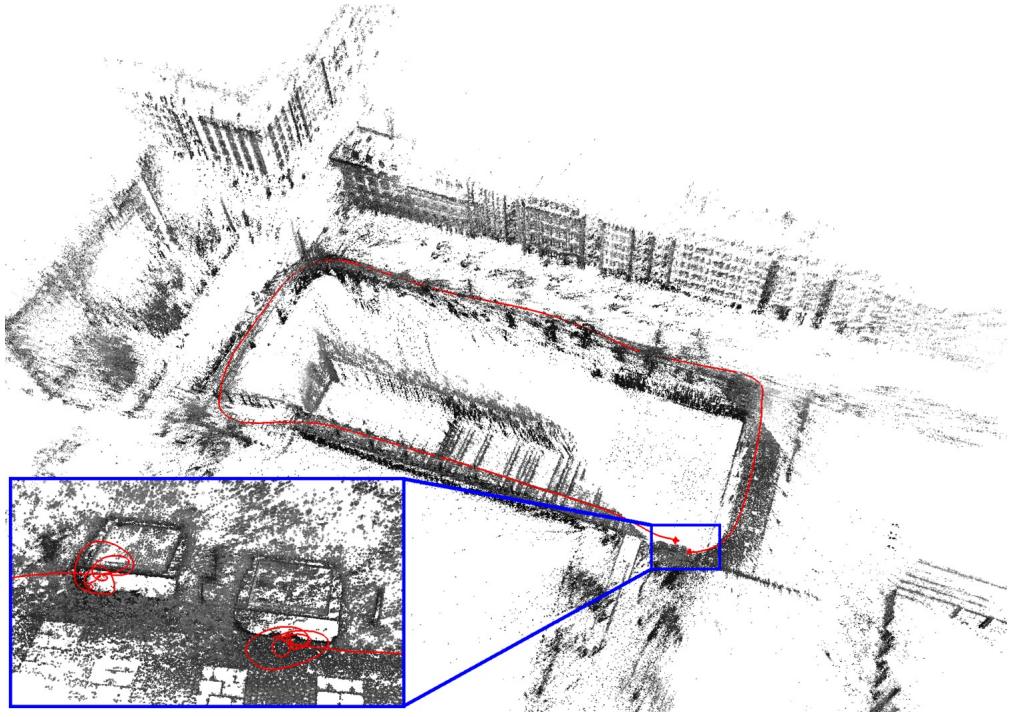


Figure 4.1: **A loop in DSO.** A whole estimated trajectory alongside the constructed three-dimensional pointcloud can be seen. The inset below on the left shows both ends of a loop, over which a considerable drift has been accumulated. Taken from [EKC17].

brightness transfer parameters addressed in Section 3.3. w_δ is the Huber weight chosen such that a squared sum of the residuals would correspond to their Huber norm with threshold δ as defined in Equation 3.7:

$$w_\delta(r) = \begin{cases} 1 & \text{if } |r| < \delta \\ \sqrt{2\,\frac{\delta}{|r|} - \frac{\delta^2}{|r|^2}} & \text{otherwise} \end{cases}$$

4.2.2 Photometric Factor

We have defined an unidirectional photometric factor in GTSAM, which pertains to the optimization of the photometric error between two given marginalized keyframes. Its energy is defined as:

$$E_{PhF}(i,j) = \sum_{\mathbf{p} \in pr(i,j)} \sum_{\mathbf{q} \in N_\mathbf{p}} w_\delta \left((I_j[\mathbf{q}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{q}] - b_i) \right) \tag{4.2}$$

We can now rewrite Equation 4.1 by incorporating photometric factors:

$$E_{global} = \sum_{(i,j)\in K} E_{PhF}(i,j) \tag{4.3}$$

4.2.3 Global Energy Optimization

We use the iSAM2 algorithm presented in [KJR+11] and implemented in GTSAM to continuously optimize the global energy presented in Equation 4.3. At its heart lies Gauss-Newton optimization similar to the one described in Section 3.6. However, there are two main differences. Firstly, we redefine the objective function to be:

$$h : (G^n)_{n\in F} \to \mathbb{R}^m \ , \ (g^n)_{n\in F} \mapsto R$$

with R – the residual vector with cardinality $m = |R|$, comprising the stacked summands of each photometric factor energy $E_{PhF}(i,j)$ as defined in Equation 4.2, whereby we use the poses in the tuple $(g^n)_{n\in F}$ to compute each residual. Secondly, use the unweighted normal equations from 3.16.

Note that we include the residuals of a photometric factor into the global optimization only if both of its associated marginalized keyframes have at least 300 marginalized points. If this is not the case, the photometric factor energy function may not be stable as a consequence of the low number of points that can be projected, resulting in an insufficient number of residuals. We therefore add a pose constraint between both frames instead, which is implemented as a Between Factor in GTSAM.

4.2.4 Partial Derivatives

In the following, we present the partial derivatives we use to compute the Jacobian matrix utilized in the normal equations. We consider a single residual, i.e summand in the global energy function 4.1:

$$w_\delta \left((I_j \left[\Pi \left(K \ T_i^j \ K^{-1} \ \Pi^{-1} \left(\mathbf{q}, d_q \right) \right) \right] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{q}] - b_i) \right)$$

Let $\nabla_{\mathbf{p}'} I_j \in \mathbb{R}^{1\times 2}$ denote the differential of the image function I_j defined in Equation 3.5 with respect to the warped point position $\mathbf{p}'$ defined in Equation 3.3. Applying the central differences along the axes x and y, we have:

$$\nabla_{\mathbf{p}'} I_j = \begin{pmatrix} \frac{\partial I_j}{\partial x} \\ \frac{\partial I_j}{\partial y} \end{pmatrix}$$

Furthermore, let $\nabla_p \Pi \in \mathbb{R}^{2\times 4}$ denote the differential of the point projection function Π defined in Equation 3.1 with respect to a point $p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ so that:

$$\nabla_p \Pi = \begin{pmatrix} \frac{1}{z} & 0 & \frac{x}{z^2} & 0 \\ 0 & \frac{1}{z} & \frac{y}{z^2} & 0 \end{pmatrix}$$

We differentiate the residual with respect to a small increment ξ_k onto T_i^j (see Equations 3.2 and 3.13):

$$\frac{w_\delta \, \partial \left((I_j \left[\Pi \left(K \, T_i^j \oplus \xi_k \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right) \right) \right] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{q}] - b_i) \right)}{\partial \, \xi_k} =$$

$$\frac{w_\delta \, \partial I_j \left[\Pi \left(K \, T_i^j \oplus \xi_k \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right) \right) \right]}{\partial \, \xi_k} =$$

$$\frac{w_\delta \, \nabla_{\mathbf{p}'} I_j \, \partial \Pi \left(K \, T_i^j \oplus \xi_k \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right) \right)}{\partial \, \xi_k} =$$

$$\frac{w_\delta \, \nabla_{\mathbf{p}'} I_j \, \nabla_p \Pi \, \partial K \, T_i^j \oplus \xi_k \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right)}{\partial \, \xi_k} =$$

$$\frac{w_\delta \, \nabla_{\mathbf{p}'} I_j \, \nabla_p \Pi \, K \, T_i^j \, \partial \, exp(\widehat{\xi_k}) \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right)}{\partial \, \xi_k} \overset{(*)}{=}$$

$$\frac{w_\delta \, \nabla_{\mathbf{p}'} I_j \, \nabla_p \Pi \, K \, T_i^j \, g_k \, \partial K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right)}{\partial \, \xi_k} =$$

$$w_\delta \, \nabla_{\mathbf{p}'} I_j \, \nabla_p \Pi \, K \, T_i^j \, g_k \, K^{-1} \, \Pi^{-1} \left(\mathbf{q}, d_q \right) =: \nabla_k$$

Note that in step (*), we have substituted $\frac{\partial \, exp(\widehat{\xi_k})}{\partial \, \xi_k}$ with the Lie group generator g_k. All six Lie generators we employ for the group $G_{SE(3)}$ are listed in Section 3.5.3.

Let $\nabla = \begin{pmatrix} \nabla_1 \\ \vdots \\ \nabla_6 \end{pmatrix}$ be the residual derivative at T_i^j. Since T_j and T_i are our optimized parameters, we have to apply the pushforwards derived in Section 3.5.7 to arrive at their respective derivatives ∇^j and ∇^i according to Equation 3.15:

$$\nabla^j = -\nabla \, Ad_{T_i^{-1} \, T_j}$$

$$\nabla^i = \nabla$$

4.3 Loop Detection and Closing

An integral part of any global optimization layer is a loop detection mechanism. In order to develop one for the sequences processed by DSO, we have employed a DBoW2 keyframe database [GLT12] of ORB feature descriptors [RRKB11] as done in ORB-SLAM2 [MT16]. We also employ the ORB vocabulary and feature extractor developed for ORB-SLAM2. Any keyframe with a sufficiently similar bag-of-words vector to the one of the currently marginalized keyframe is considered a loop closing candidate. By matching the descriptors of both images taken at the two keyframes, we obtain feature correspondences and subsequently perform RANSAC [FB81] to filter out 8 point correspondences free from outliers with a probability of 99% as input for the normalized eight-point algorithm, which provides an estimate of the fundamental matrix (see Section 3.7) of the transformation between both keyframes. This pipeline is inspired by [MAT14]. We then use the rotation estimates R and R^T extracted from the fundamental matrix as outlined in Section 9.6 of [HZ04] as an initialization for a Levenberg-Marquardt optimization of the $SE(3)$ transformation between both keyframes as defined in Equation 3.18. We have found the translation estimate to be highly unreliable because of the intrinsic scale ambiguity of its extraction from the fundamental matrix and therefore leave it out of the initialization. This initialization works even in case of huge rotational drift of the DSO estimates, when an accurate $SE(3)$ transformation cannot be obtained by optimizing the relative transformation between both keyframes arising from the DSO estimates of their poses by means of Equation 3.2.

We minimize the photometric error in projections of the current keyframe's sample points as found by DSO into the loop closing candidate keyframe and in the opposite direction, while keeping estimated point depths fixed, and thus compute an $SE(3)$ transformation each way between the keyframes. Since only two keyframes are considered at this step, we are not affected by the scale drift that might have been accumulated throughout the trajectory. We wouldn't therefore benefit from the computation of a $Sim(3)$ transformation as done in LSD-SLAM upon computation of pose constraints originating from loop closures.

We validate the two $SE(3)$ transformation estimates for arriving from one end of the loop to its opposing one by imposing several conditions for geometric consistency:

- both $SE(3)$ transformations must yield at least 300 point projections for short loops, i.e. loops, in which both keyframes are at most 100 keyframes apart, or 100 point projections for long loops, i.e. loops, in which both keyframes are more than 100 keyframes apart

- the angle of the rotational part of both $SE(3)$ transformations in angle-axis representation may not exceed $\Pi/2$ radians

- the angle of the rotational part of the error transformation in angle-axis representation may not exceed 0.1 radians
- the squared norm of the translational part of the error transformation may not exceed 0.001

The error transformation $T_1\ T_2$ is computed as the composition, i.e. matrix multiplication, of the transformation estimates T_1 from the currently marginalized keyframe to the loop closing candidate and T_2 in the opposite direction. Note that we only consider sufficiently long loops, i.e. ones that stretch over more than 100 keyframes, as we have found DSO resilient enough to not accumulate any substantial drift over the course of smaller loops. An evaluation of the accuracy and the recall of our loop detection routine is presented in Section 5.2.

We have also tried out taking into account other error metrics such as the energy of point projections as defined in Section 4.2.2 or the difference, correlation or normalized correlation between patches around sample points and patches around their projections. We could not, however, discern any notable improvements.

The $SE(3)$ transformations obtained by our loop closing routine that have withstood a geometric consistency check are included as pose constraints into the global optimization layer. For the implementation of a pose constraint, GTSAM provides the **BetweenConstraint** subclass of **BetweenFactor**, which imposes a huge penalty for any deviations from its initialization.

4.4 Integration into DSO

The flowchart in Figure 4.2 depicts the integration of our global optimization layer into DSO. Several of its fields are broken down into further algorithm steps in subsequent figures. The 'Add GTSAM factors' field is expanded in Figure 4.3, the 'Are there any loop detections?' field – in Figure 4.4 and the 'Global optimization' field – in Figure 4.5.

The essence of the original DSO algorithm can be summarized in several paramount steps. First comes the creation of frames and the sampling of points. Then a frame gets added to the Local Optimization Window (LOW) whenever a new keyframe is needed. This has the marginalization of sample points and keyframes in the LOW as a consequence so that the optimization problem remains sparse. Subsequently the LOW is optimized by means of the Gauss-Newton algorithm and DSO proceeds to create new frames from the image sequence.

Our global optimization layer commences with the marginalization of a keyframe from the LOW. At that point, its pose and all of its sample point depths are fixed and no residuals in the local optimization window energy from Equation 3.8 depend on it anymore. We can therefore use its current pose estimate as an initialization for the pose of the keyframe variable in the factor graph. Furthermore, we define factors that depend on this newly introduced variable. A Prior Factor

serves as an anchor for the whole optimization by enforcing the coincidence of the first keyframe pose with the origin of the coordinate system. Our custom Photometric Factors, whose energy is defined in Equation 4.2, pertain to a global revision of the local optimization window energy from Equation 3.8. They depend on two keyframe variables, each referring to a marginalized keyframe, so that both keyframes have belonged to the same LOW at any point in time. Their relative pose defined in Equation 3.2 is optimized unidirectionally. As a guarantee for the stability of this optimization, we need to ensure that enough points can be projected between both keyframes. We therefore impose two conditions. Firstly, each of the keyframes has to possess at least 300 interest points. Secondly, the projected image plane overlap shown in Figure 4.6 has to surpass a given threshold in either direction. It serves as an estimate for the amount of points that can be projected between both cameras with a given transformation and is computed by means of the polygon clipping library Polyclipping, which is based on the polygon clipping algorithm presented in [Vat92] and can compute the intersection of arbitrary polygons in two-dimensional space. While even an overlap as small as 30% suffices for most sequences, we use a threshold of 60% for enhanced robustness and computational efficiency. If any of these conditions is not fulfilled, we are not able to further improve the DSO estimate for the relative pose between both keyframes and therefore add Between Factors to the global factor graph ensuring that their relative pose will not move far away from its initialization.

Beyond these factors, we also employ pose constraints for aligning both ends of detected loops. To this end, we compute the feature descriptors of the currently marginalized keyframe and compare them against the feature descriptors of other already marginalized keyframes stored in a keyframe database. We subsequently estimate a fundamental matrix for any of the best matches and the currently marginalized keyframe if they have not shared the same LOW at any point in time as described in Section 4.3. We then optimize an $SE(3)$ transformation either way between the keyframes initialized as the respective rotation estimate extracted from the fundamental matrix. If the estimated $SE(3)$ transformations satisfy the conditions for geometric consistency listed in Section 4.3, we add a Between Constraint to the global factor graph that imposes these relative transformations between the keyframes, at which the loop was detected. The Photometric and Between Factors added at previous steps ensure that the alignment caused by the Between Constraint is spread over the whole trajectory in order to reconstruct a globally consistent robot motion. Note that we have experimented with the substitution of these Between Constraints with Photometric Factors, but they were not able to counteract the accumulated drift over the course of the loop at the finest level of the Gaussian pyramid. We have also tried to perform this substitution at successive optimization steps, but could not ascertain any further optimization. A broader discussion of possible improvements can be found in Section 6.2

We perform global factor graph optimization whenever new factors or a Between Constraint have been added under the condition that a Prior Factor is present. The global optimization problem is otherwise underconstrained.

Once the whole sequence is processed, we intercept DSO once again to perform a finalization step, in which all sample points and all active keyframes get marginalized. This brings about the processing of each marginalized frame by our global optimization layer as described above. Once the finalization step comes to an end, we also reach the end of our algorithm.

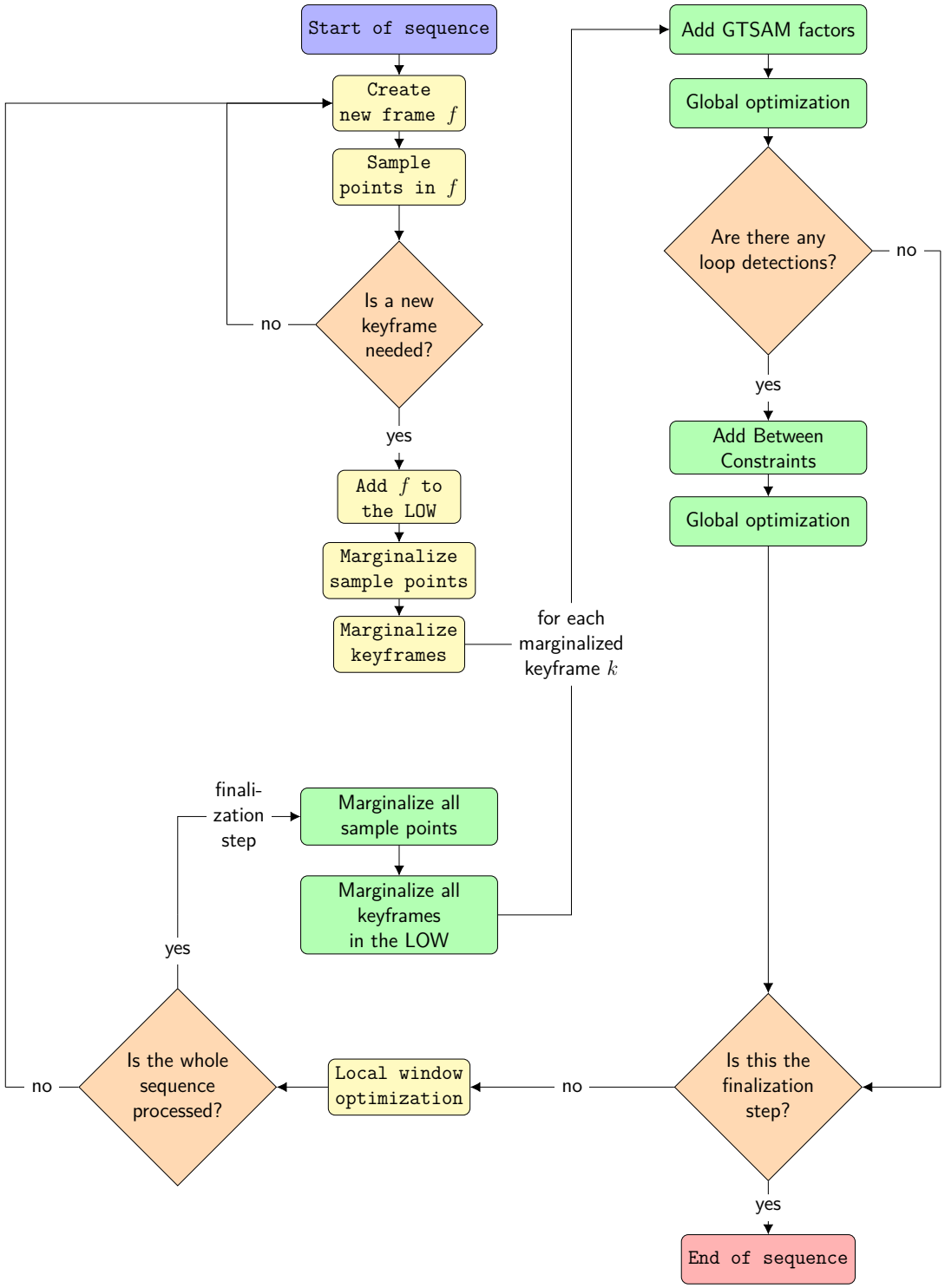


Figure 4.2: **Flowchart depicting the integration of our global optimization layer into DSO.** The actions on yellow background comprise the original DSO algorithm. It has been stripped down to the steps relevant for global optimization. The green background designates actions in the global optimization layer. Some of the fields of this flowchart are expanded in Figures 4.3 – 4.5 to show further steps in more detail.

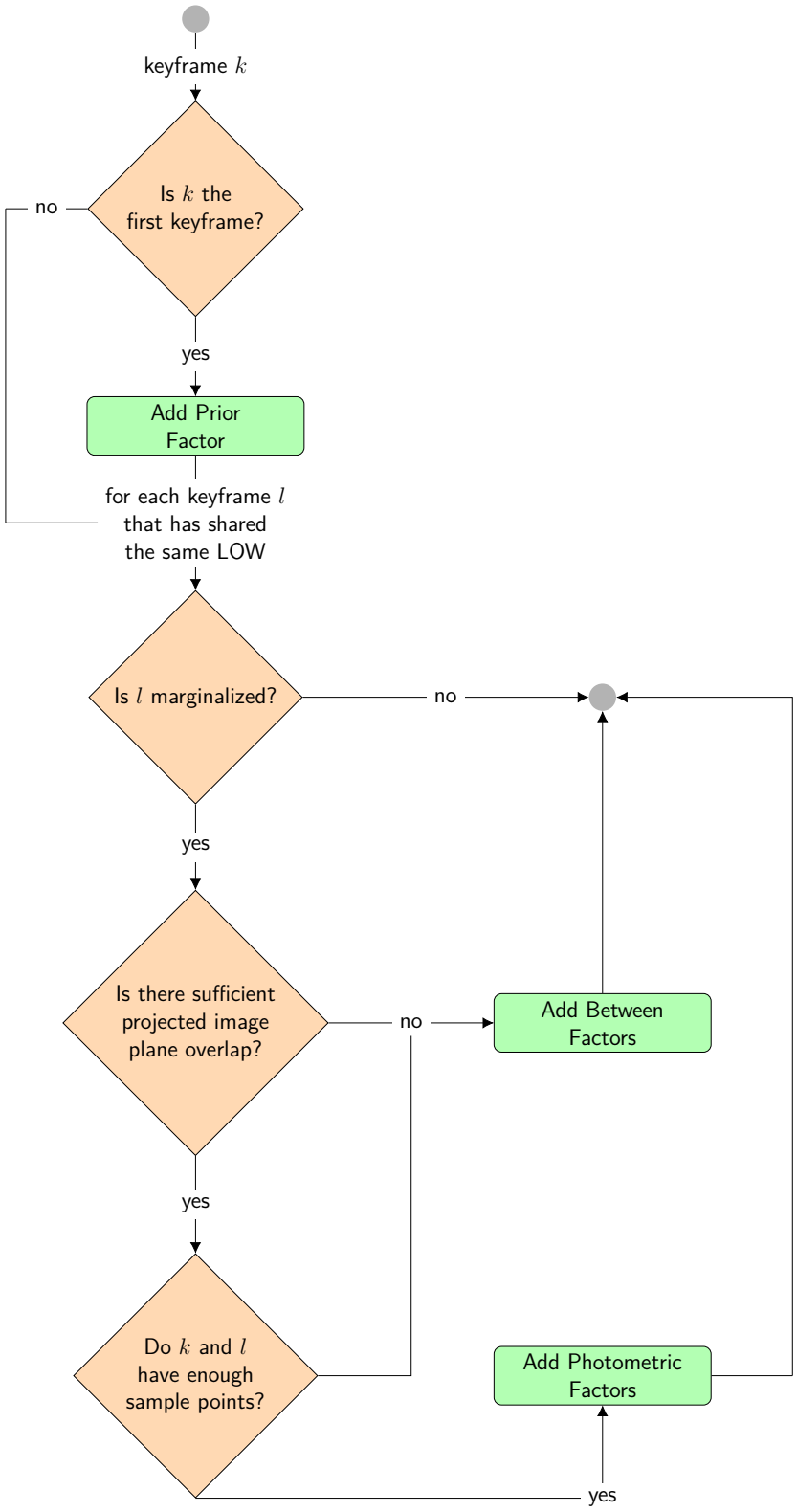


Figure 4.3: **Add GTSAM factors**

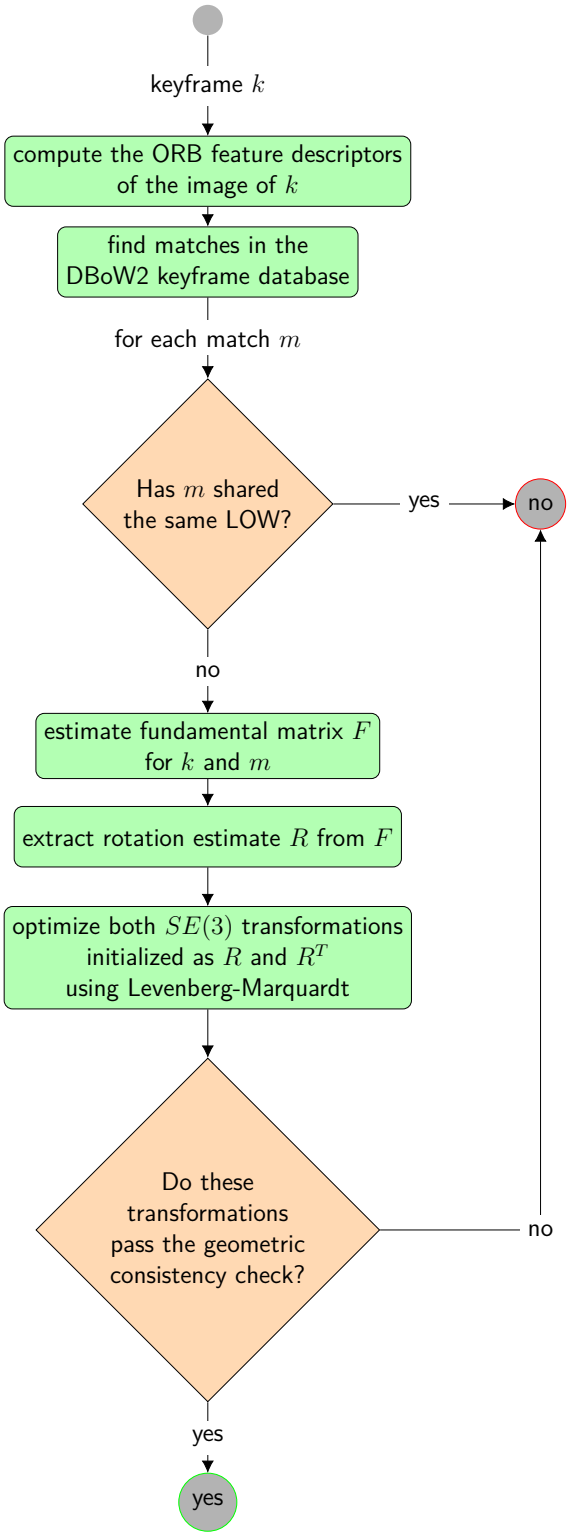


Figure 4.4: **Are there any loop detections?**

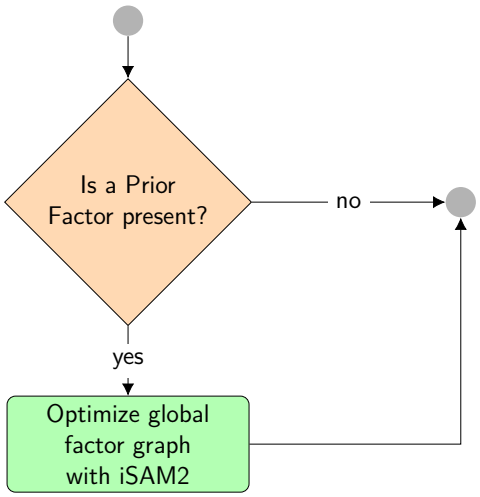


Figure 4.5: **Global optimization**

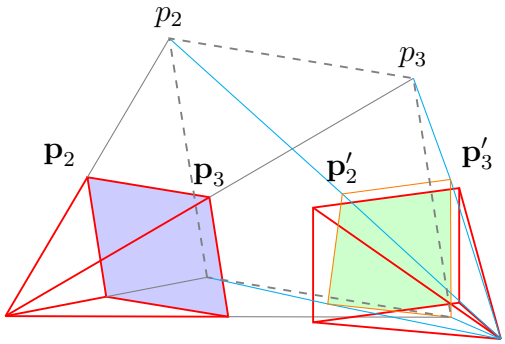


Figure 4.6: **Projected image plane overlap**. Let Π, K and T_i^j be defined as in Section 3.2. Let the points $\mathbf{p}_1$ to $\mathbf{p}_4$ lie at the corners of the image plane of the reference frame i drawn in blue and their estimated corresponding three-dimensional points be defined as $p_k = \Pi^{-1}(\mathbf{p}_k, d_{m_i})\ \forall k \in [1, ..., 4]$, where d_{m_i} stands for the median inverse depth of all sample points in i. The estimated projections of these points into the target frame j are then defined as $\mathbf{p}'_k = \Pi\ (K\ T_i^j\ K^{-1}\ p_k)\ \forall k \in [1, ..., 4]$. The projected image plane overlap drawn in green is the intersection area of the image plane of the target frame with the polygon defined by the points $\mathbf{p}'_k\ \forall k \in [1, ..., 4]$.

Below, we present the pseudocode of the stripped-down version of *DSO* with integrated global optimization layer shown in the flowchart from Figure 4.2 . *dso* is an object representing the full DSO system with some self-explanatory functions for creating and marginalizing frames and sample points, as well as for local window optimization. *low* is the local optimization window object, fg - the global factor graph object and $isam2$ – the iSAM2 optimizer with associated function $globally\_optimize(fg)$. $end\_of\_seq()$ returns true as soon as the end of the processed sequence is reached.

Let k and l be marginalized keyframes. The procedure `ProcessMarginalizedFrame`(k), by means of which our global optimization layer handles marginalized frames, is defined in Algorithm 2. **PriorFactor**(k), **PhotometricFactor**(k, l), **BetweenFactor**(k, l) and **BetweenConstraint**(k, l) refer to GTSAM classes implementing the respective factors. $slow(k)$ returns a set of keyframes, which have shared the same LOW as k at any point in time. $pipo(k, l)$ computes the projected image plane overlap at median sample point inverse depth as shown in Figure 4.6 using the polygon clipping library Polyclipping. θ_{pipo} stands for the threshold we impose on this overlap in order to add photometric factors. $nep(k)$ returns true if keyframe k possesses less than 300 sample points.

We also define the following functions employed for loop detection. $orb(k)$ returns a vector fd of descriptors of the ORB features extracted from the image of k, whose keyframe database matches are retrieved with the function $dbow2\_matches(fd)$. $fund\_mat(k, m)$ estimates the fundamental matrix fm for k and m as described in Section 4.3. The rotation estimate rot extracted with $extract\_rotation(fm)$ and its inverse $transpose(rot)$ are then used in $se3(k, m, rot)$ and $se3(m, k, transpose(rot))$, respectively, as an initialization for the estimation of $SE(3)$ transformations $trKtoM$ from k to m and $trMtoK$ from m to k. Their geometric consistency is checked by the function $geo\_consistent(trKtoM, trMtoK)$.

The functions marginalize_all_sample_points() and marginalize_keyframe(k) refer to the marginalization of all sample points and active keyframes in the finalization step of our algorithm.

Algorithm 1 Integrate global optimization layer into DSO

```
1: procedure DSOwithGOL
2:     while !end_of_seq() do
3:         dso.create_frame(f);
4:         dso.sample_points(f);
5:         if new keyframe needed then
6:             low.add(f);
7:             dso.marginalize_sample_points();
8:             mkfs = dso.marginalize_keyframes(low);
9:             for k ∈ mkfs do
10:                ProcessMarginalizedFrame(k);
11:            end for
12:            dso.locally_optimize(low)
13:        end if
14:    end while
15:    marginalize_all_sample_points();
16:    for k ∈ low do
17:        marginalize_keyframe(k);
18:        ProcessMarginalizedFrame(k);
19:    end for
20: end procedure
```

Algorithm 2 Add GTSAM factors and constraints for a given marginalized frame whenever needed and perform global factor graph optimization

procedure ProcessMarginalizedFrame(k)
  if k is the first keyframe **then**
    $fg.add($**PriorFactor**$(k))$;
  end if
  for $l \in slow(k)$ **do**
    if l is not marginalized **then**
      continue;
    end if
    if $pipo(k,l) < \theta_{pipo}$ **or** $nep(k)$ **or** $nep(l)$ **then**
      $fg.add($**BetweenFactor**$(k,l))$;
      $fg.add($**BetweenFactor**$(l,k))$;
    else
      $fg.add($**PhotometricFactor**$(k,l))$;
      $fg.add($**PhotometricFactor**$(l,k))$;
    end if
  end for
  $isam2.globally\_optimize(fg)$
  $fd = orb(k)$
  for $m \in dbow2\_matches(fd)$ **do**
    if $m \in slow(k)$ **then**
      continue;
    end if
    $fm = fund\_mat(k,m)$
    $rot = extract\_rotation(fm)$
    $trKtoM = se3(k,m,rot)$
    $trMtoK = se3(m,k,transpose(rot))$
    if $geo\_consistent(trKtoM, trMtoK)$ **then**
      $fg.add($**BetweenConstraint**$(k,m))$;
      $isam2.globally\_optimize(fg)$
    end if
  end for
end procedure

5 Evaluation

We evaluate the accuracy of the $SE(3)$ pose estimates at the keyframes created by DSO and globally optimized by our approach described in Chapter 4. To this end, we compare and contrast the optimized trajectories with the trajectories estimated by the original DSO algorithm and a reduced version of ORB-SLAM with disabled explicit loop-closure detection and relocalization taken from the supplementary material of [EKC17]. We have excluded LSD-SLAM from the evaluation since it reportedly fails consistently on most sequences from the dataset described in the next section as stated in Section 4 of [EKC17].

5.1 Dataset and Evaluation Metrics

The TUM monocular visual odometry dataset [EUC16] served as a benchmark for the evaluation of our method against DSO and ORB-SLAM. It comprises 50 sequences taken in both indoor and outdoor environments, all of which are photometrically calibrated so that the exposure time of each frame, the camera response function and the vignetting are recorded. Ground truth poses for time slices at the beginning and at the end of each sequence obtained from a motion capture device can be found in the supplementary material of [EKC17]. We have aligned the pose estimates delivered by each algorithm with these ground truth poses in order to compute the Absolute Trajectory Error (ATE). We have also computed the Relative Pose Error (RPE) in reference to the available ground truth poses. Both error measurements are defined below and discussed in more detail in [SEE+12].

Let $(P_i)_{i\in[1,\dots,n]}$ be the $SE(3)$ pose estimates delivered by a tracking algorithm, which have already been synchronized with the ground truth poses $(Q_i)_{i\in[1,\dots,n]}$. The RPE is meant to assess the drift over the course of the estimated trajectory. It first computes the relative pose errors for each pair of poses lying Δ timesteps apart:

$$E_i = (Q_i^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta})$$

and the root mean squared error of their translational components $trans(E_i)$:

$$RMSE_\Delta = \sqrt{\frac{1}{n-\Delta} \sum_{i=1}^{n-\Delta} \|trans(E_i)\|^2}$$

The RPE equals the average of these errors over all possible time intervals Δ:

$$RPE = \frac{1}{n} \sum_{\Delta=1}^{n} RMSE_\Delta$$

For the computation of the ATE, we first need to estimate a transformation S mapping the estimated onto the ground-truth trajectory, e.g. by application of the method presented in [Hor87]. We then take the root mean squared error of the translational components of the differences between the ground-truth and the mapped estimated trajectory at each timestep:

$$ATE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \|trans(Q_i^{-1} S P_i)\|^2}$$

Note that we have not employed the error evaluation metrics proposed in Section 3 of [EUC16], as the authors have explicitly pointed out that they do not apply to methods incorporating loop closures. Since they play a pivotal role in our approach, an evaluation of the global optimization with disabled loop detection would not have made any sense.

5.2 Loop Detection

All sequences in the TUM monocular visual odometry dataset start and end at the same three-dimensional position. We have therefore visually evaluated the accuracy and precision of our loop detection mechanism in reference to its ability to find the long loops stretching from the beginning to the end of each sequence in the dataset. Thereby, we did not take into account the number of loop detections pro sequence, whose mean is 118, but checked whether there is at least one long loop detection for each sequence to evaluate the recall and whether all loop detections in the sequence were correct for the evaluation of precision. Our loop detection routine has a recall of 84% as it has repetitively found a long loop in 42 out of the 50 sequences in the TUM monocular visual odometry dataset. Its precision lies at 93% as the loops detected in 39 out of those 42 sequences were correct.

Since we do not have ground truth about all absolute keyframe poses, we have relied on the visualization of point projections between the two keyframes at both ends of the loop to assess the quality of the $SE(3)$ transformation estimate. Some

qualitative results can be seen in Figure 5.1. Both Figure 5.1d and Figure 5.1e show loop detections from the same sequence, the second of which is a false one. This can be attributed to the repetitive structure of the windows of the depicted building.

Note that our loop detection routine works both outdoors and indoors. In the absence of repetitive structures, it provides reliable $SE(3)$ transformation estimates for loop closing even if there is a substantial rotational or translational offset or a big scale difference as long as enough sample points can be projected from the reference into the target frame upon initialization and during the Gauss-Newton optimization.

5.3 Runtime Evaluation

We have measured the runtime of all optimization layer steps shown in the flowchart in Figure 4.2. The addition of GTSAM factors takes on average 637 μs, the following global optimization step – 335 ms in case no Between Constraints have been added to the global factor graph yet and $2,3$ s in the presence of at least one Between Constraint. The loop detection lasts on average 29 ms and the global optimization just after a loop has been detected and its corresponding Between Constraint has been added to global factor graph – $7,3$ s. The runtimes provided for the global optimization steps encompass all Gauss-Newton iterations until convergence. All of the remaining steps are completed in negligible time.

5.4 Global Optimization

Our global optimization manages to recover even from significantly bad pose estimates as an initialization. In one of the test runs, the reduction of the translational part of the relative pose error between the first and the last keyframe has amounted to a factor greater than 90 for a translation norm reduction from $10,41$ to $0,114$.

5.4.1 Quantitative Comparison with DSO and ORB-SLAM

The results we obtained using the error metrics defined in Section 5.1 are summarized in the cumulative distribution functions plotted in Figure 5.2. It is worth noting that our approach significantly outperforms DSO and the reduced version of ORB-SLAM with respect to both error metrics. Its ATE remains below $0,4$ in 56% of the sequences, below 1 in 72% of the sequences and does not exceed 2 in 90% of the sequences. The respective percentages for DSO are 22%, 50% and 72%, while ORB-SLAM only manages to achieve 2%, 22% and 56%. Our approach thus clearly has the most globally consistent trajectory estimates. Its local consistency is also remarkable with 66% of the sequences yielding an RPE below 1 in comparison to only 44% for DSO and 24% for ORB-SLAM.

5.4.2 Qualitative Results

In Figure 5.3, one can see all the factors we add to the global factor graph and the resulting optimized trajectory for sequence 6. Note the big transformation between some pairs of keyframes, for which a Between Factor has been added to the factor graph. Their corresponding lines are drawn in cyan such as the ones enclosed in a red ellipse in Figure 5.4c. Both keyframes have shared the same local optimization window. We could not, however, guarantee the stability of a photometric optimization of their relative pose because of the small amount of points that can be projected with such a big translational and rotational offset. In order to sufficiently constrain all keyframe variables in the factor graph, we have instead added Between Factors meant to maintain the relative pose computed from the DSO keyframe pose estimates.

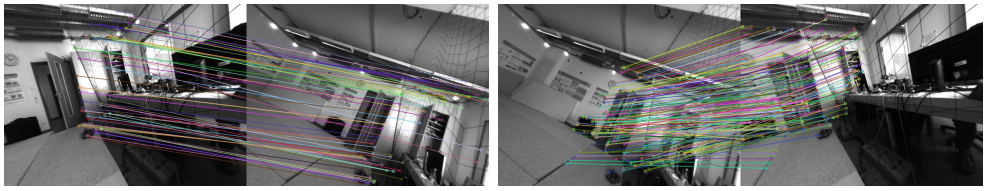
In the optimized trajectory, note the alignment of pairs of keyframes, for which a loop has been detected and therefore a Between Constraint has been added to the factor graph. Such keyframe pairs are connected with a magenta edge in Figures 5.3 and 5.4. The Between Constraints have imposed the transformation estimates by our loop detection mechanism in order to correct the motion drift over the course of the loop and bring global consistency. The Photometric and Between Factors have meanwhile ensured the local consistency of the optimized trajectory.

Among the sequences shown in Figures 5.5 and 5.6, our global optimization fails only in case of a false loop detection as apparent for sequence 34 depicted in

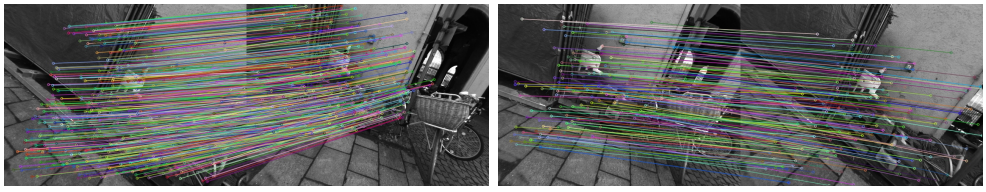
Figure 5.5d. For all other sequences, it manages to deliver an accurate trajectory estimate even if faced with a substantial motion drift.

The pointcloud visualizations in Figure 5.7 also help us to assess the quality of a trajectory estimate. Since all marginalized sampled points from all keyframes are shown, we can recognize the motion drift accumulated over the course of a loop by the discrepancy of the points originating from different keyframes, but representing the same object. The fusion of these shifted object representations at both ends of the loop visible in the optimized pointcloud is the result of the accumulated motion drift correction due to our global optimization incorporating loop closures.

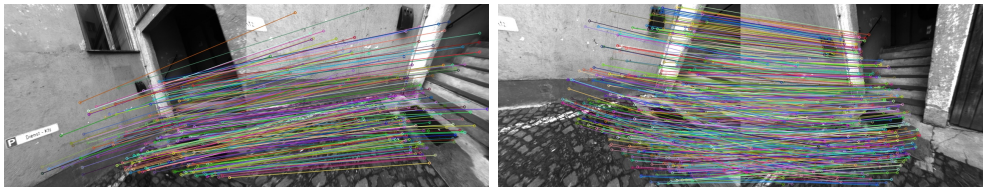
Figure 5.8 gives us some further examples of pointcloud fusion. The accumulated drift over the course of a long loop often makes DSO 'see double' Note the vast translational offset between drawings of the same object in the pointclouds delivered by DSO and its elimination in the pointclouds rendered at the globally optimized keyframe poses. In Figure 5.8a, the bike ramp drawn on the far left is the same as the one drawn on the far right. Both drawings are fused in Figure 5.8b owing to the aligned keyframe poses in the optimized trajectory, from which the sample points represented in the pointcloud originate. Both bike drawings in Figure 5.8c have been likewise fused in Figure 5.8d.



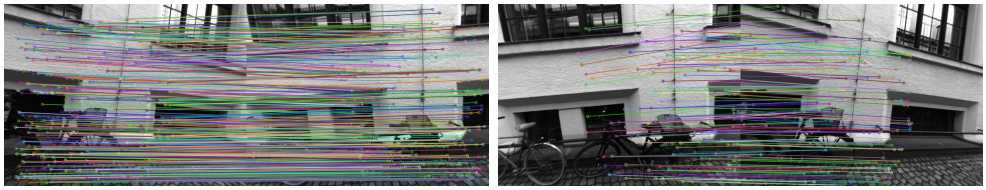
(a) Sequence 9



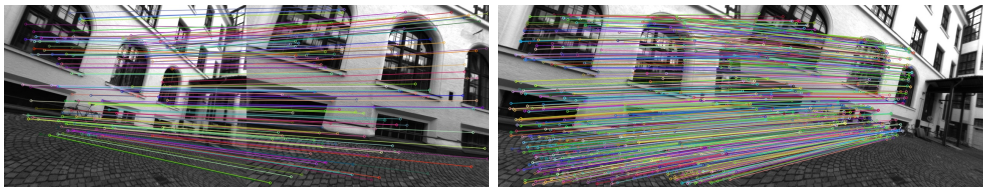
(b) Sequence 30



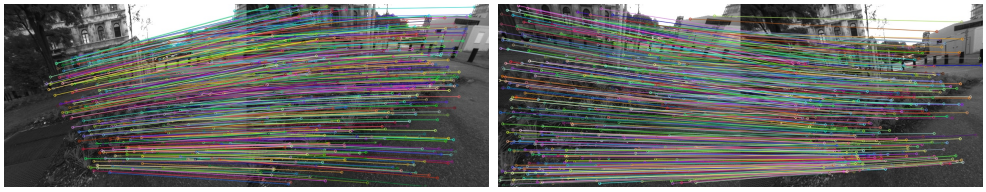
(c) Sequence 33



(d) Sequence 34



(e) Sequence 34



(f) Sequence 50

Figure 5.1: **Loop detections.** As two $SE(3)$ transformations have been estimated unidirectionally and independently by our loop closing routine, we visualize each of them next to each other in the same row.

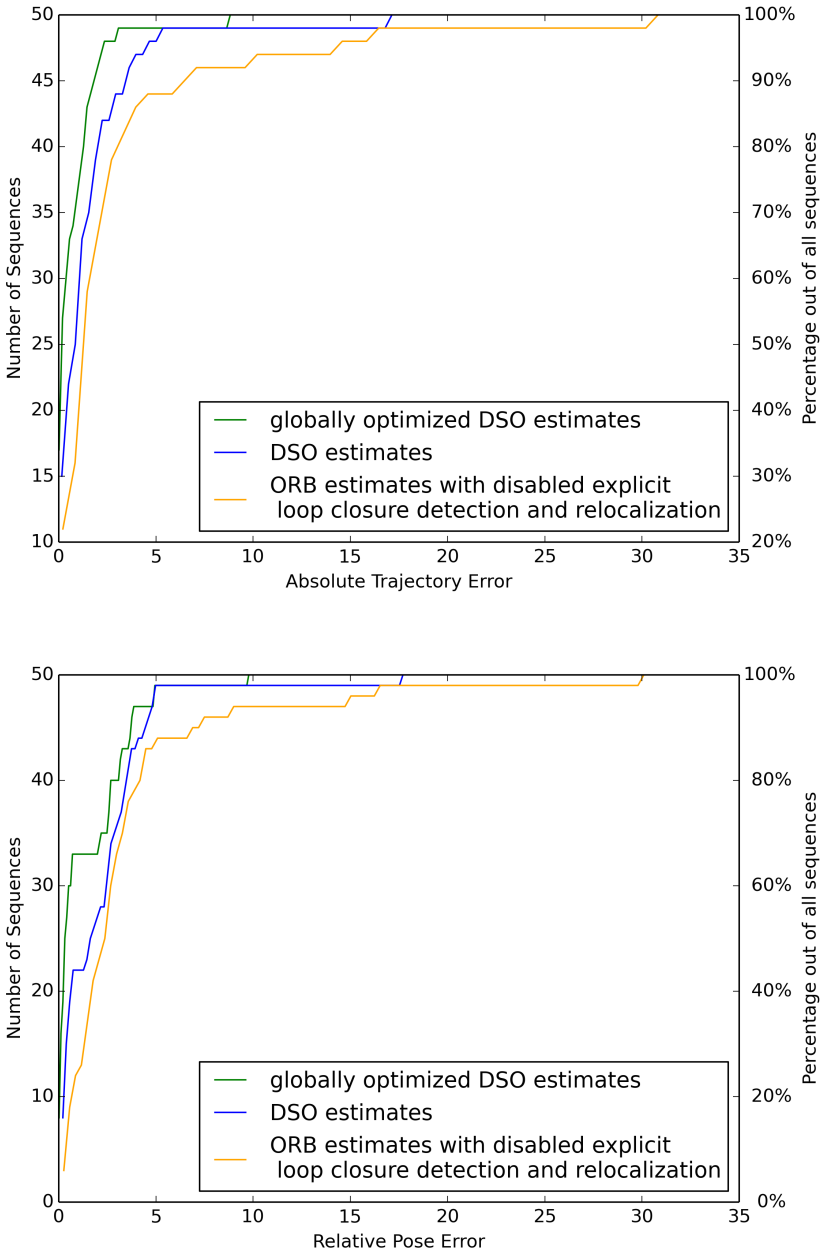
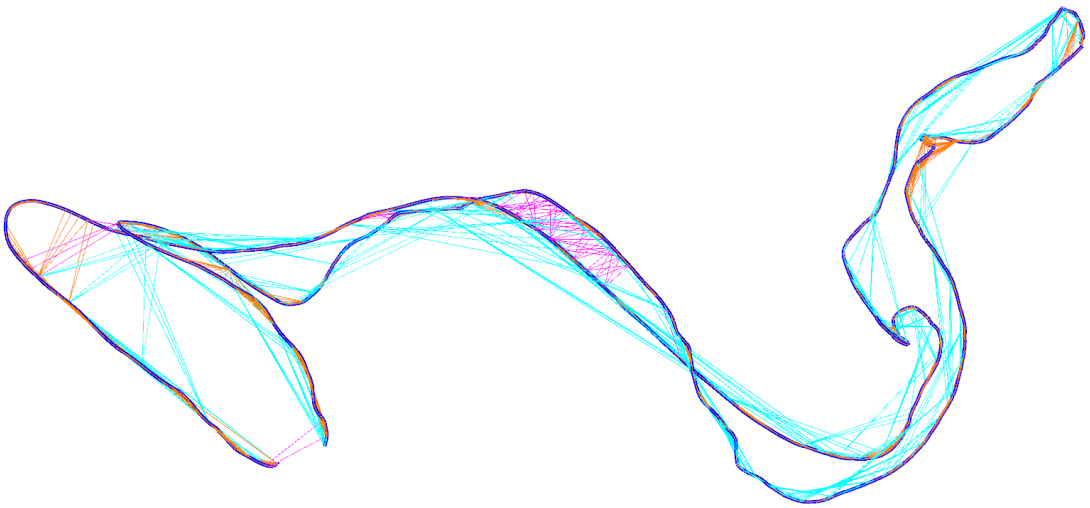


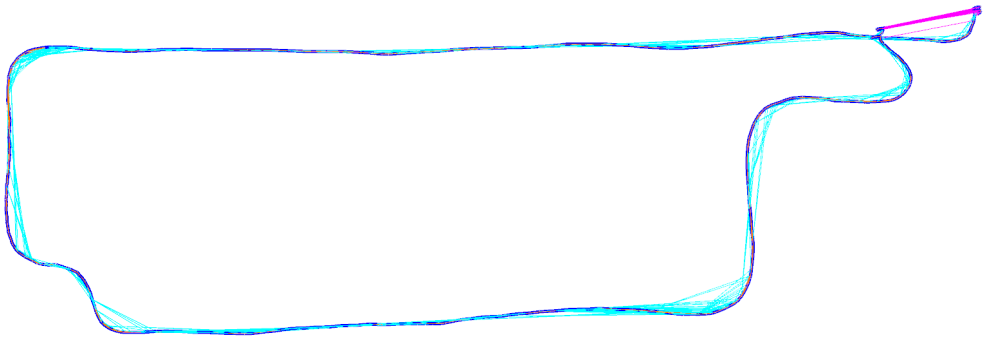
Figure 5.2: **Cumulative distribution functions of the absolute trajectory and the relative pose error of our approach, DSO and ORB-SLAM evaluated on the TUM monocular visual odometry dataset.** To each error value on the x-axis corresponds the number of sequences on the y-axis, for which the error was smaller. For each sequence , we have taken the median error over several test runs in order to eliminate the influence of outliers and thus do justice to the intrinsic non-determinism of the evaluated systems.



(a) DSO trajectory estimate and GTSAM factors for sequence 6

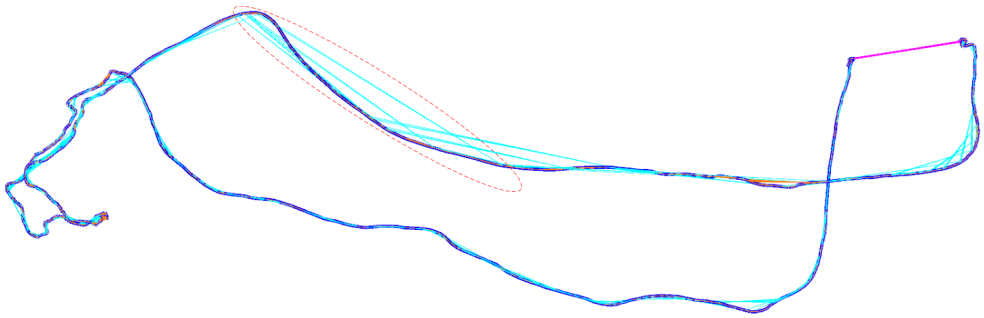
(b) Optimized trajectory estimate for sequence 6

Figure 5.3: **GTSAM factors and global optimization.** The trajectory estimate of DSO is shown in blue and the globally optimized one – in green. GTSAM factors are visualized as lines connecting pairs of keyframes on which they depend. Photometric factors are drawn in orange, Between Factors – in cyan and Between Constraints – in magenta. The GTSAM factors, as well as the optimized trajectory of two more sequences are shown in Figure 5.4.



(a) DSO trajectory estimate and GTSAM factors for sequence 17

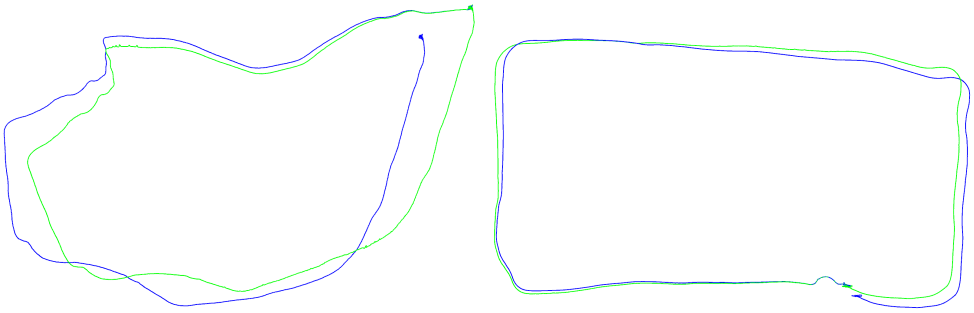
(b) Optimized trajectory estimate for sequence 17



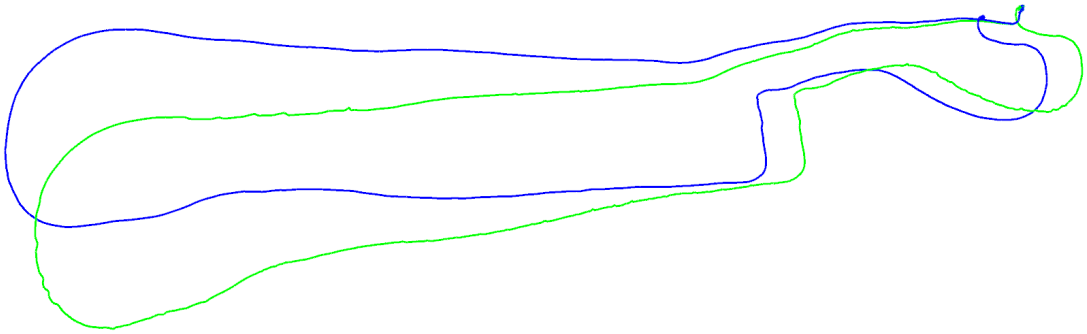
(c) DSO trajectory estimate and GTSAM factors for sequence 22

(d) Optimized trajectory estimate for sequence 22

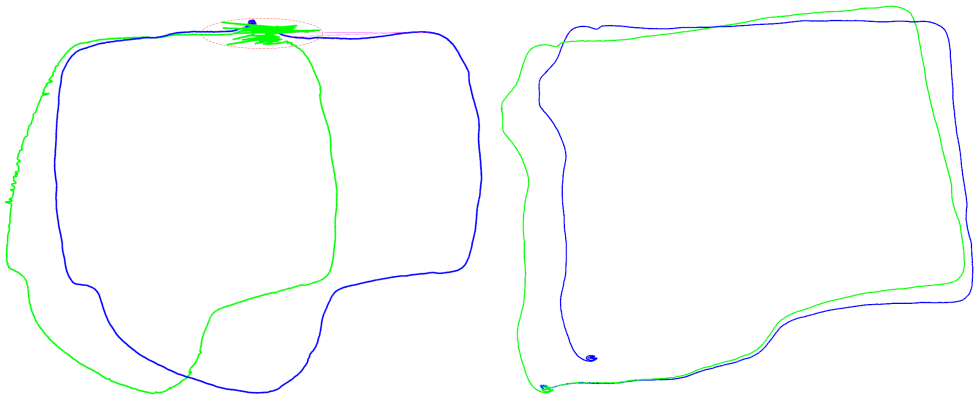
Figure 5.4: **GTSAM factors and global optimization.**



(a) Sequence 21 (b) Sequence 50

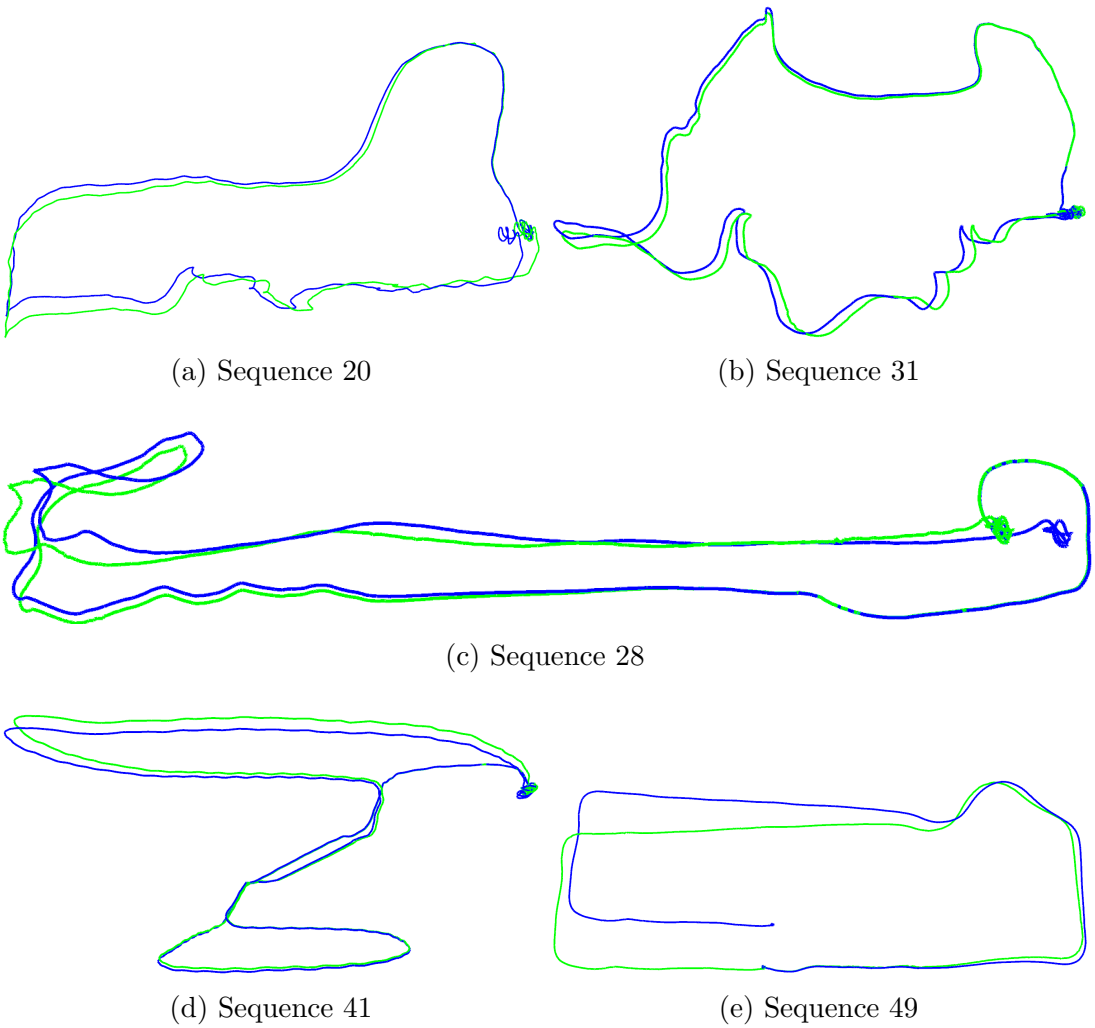


(c) Sequence 18



(d) Sequence 34 (e) Sequence 19

Figure 5.5: **Initial and globally optimized trajectories.** The trajectory in blue represents the DSO keyframe pose estimates obtained by local window optimization. The green trajectory is the result of our global optimization approach initialized with the DSO keyframe pose estimates from the blue trajectory. Note that the green trajectory at the bottom left corner is distorted as a consequence of the false loop detection shown in the last row of Figure 5.1. The zigzag pattern enclosed in a red ellipse corresponds to the part of the original trajectory between both keyframes, for which a false loop has been detected. More trajectories build from initial and globally optimized DSO keyframe pose estimates can be seen in Figure 5.6.

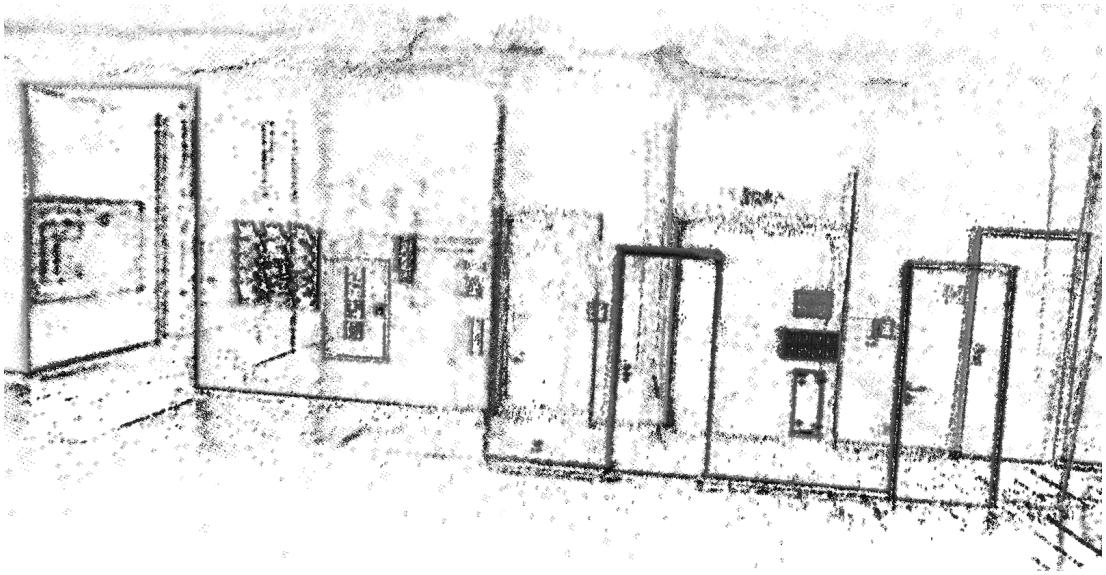


(a) Sequence 20 (b) Sequence 31

(c) Sequence 28

(d) Sequence 41 (e) Sequence 49

Figure 5.6: **Initial and globally optimized trajectories.** The trajectory drawn in blue is build from DSO keyframe pose estimates obtained by local window optimization, whereas the trajectory drawn in green – from the globally optimized keyframe pose estimates.



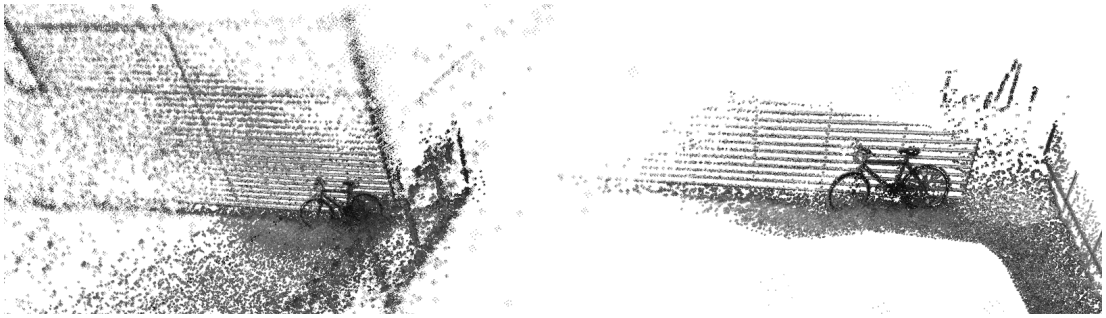
(a) Part of the pointcloud delivered by DSO for sequence 18

(b) Corresponding part of the optimized pointcloud delivered by our approach for sequence 18

Figure 5.7: **Pointclouds before and after global optimization.** Note the offsets of the drawings of the same doors in the image above and their fusion in the image below. Initial and optimized pointclouds drawn while processing two more sequences are shown in Figure 5.8.

(a) Part of the pointcloud delivered by DSO for sequence 21

(b) Corresponding part of the optimized pointcloud delivered by our approach for sequence 21



(c) Part of the pointcloud delivered by DSO for sequence 22

(d) Corresponding part of the optimized pointcloud delivered by our approach for sequence 22

Figure 5.8: **Pointclouds before and after global optimization.**

6 Conclusion and Future Work

6.1 Final Remarks

The global optimization layer we have developed for DSO has accomplished a substantial motion drift reduction in the estimated trajectory. We have successfully implemented our direct global energy formulation in a factor graph by means of custom Photometric Factors. In the absence of enough point projections, we have instead added Between Factors for pairs of keyframes that have belonged to the same local optimization window in order to sufficiently constrain their pose variables in the factor graph.

While the Photometric and Between Factors served for local consistency, we have also enforced Between Constraints for pairs of keyframes, between which a loop has been detected, so as to achieve global consistency of the estimated trajectory. The loop detection mechanism we have devised is based on matching the feature descriptors of the image taken at the current keyframe against a bag-of-words keyframe database, estimating an $SE(3)$ transformation between the current keyframe and any sufficiently good database match and geometrically verifying it in order to filter out correct loop detections.

The evaluation of our loop detection routine has shown its accuracy and robustness on the TUM monocular visual odometry dataset. Our global optimization framework evaluation on the same dataset has demonstrated a considerable improvement of the original DSO algorithm in terms of the absolute trajectory and the relative pose error. It has also significantly outperformed a reduced version of ORB-SLAM, which was designed for assessment against visual odometry approaches. The qualitative results have further convinced us of the merits of our global optimization.

6.2 Future Work

We have noticed that our approach can be developed further to fully release its potential and therefore propose several research directions building upon this thesis.

The global optimization of both keyframe camera poses and sample point depths can be considered as a next step in the development of our approach. Point depths can be added to the optimized parameters and the partial derivatives of the residuals with respect to point depths appended as an additional column to the Jacobian matrix of the objective function. As an alternative, photometric factors may be redefined so that each of them optimizes the residual of a single point projection. Each point depth will then be represented by its own variable in the factor graph. This may, however, impede the global optimization altogether as the number of factors will grow substantially.

A pure photometric optimization may be added as a final step of the global optimization. We have, however, noticed that a simple replacement of the pose constraints for keyframes at both ends of a detected loop with photometric factors at the finest Gaussian pyramid level does not lead to any further optimization. We have also found out that the global optimization runs out of memory if we perform a coarse-to-fine optimization of the photometric factors. This problem could be circumvented by sequential optimization of incremental factor graphs at each Gaussian pyramid level. The sequential addition of new keyframe pose variables at each step can for example emulate the sequential marginalization of keyframes in DSO.

Our approach significantly reduces the motion drift accumulated over the course of the trajectory. Another problem with monocular visual odometry is the inherent scale ambiguity of the observed environment. To counteract a possible scale drift, three-dimensional similarity instead of Euclidean transforms can be used, which incorporate an additional degree of freedom for scale.

The development of our approach into a full SLAM system requires a global relocalization mechanism in case of lost tracking. It would also profit from a global routine for detecting and purging redundant keyframes to ensure lifelong operation.

A parallelization of DSO with integrated global optimization can reduce the overall computational time. The global optimization layer can be placed into a separate thread, running simultaneously with the main one executing DSO.

Bibliography

[Ago05] M.K. Agoston. *Computer Graphics and Geometric Modelling*. Computer Graphics and Geometric Modeling. Springer, 2005.

[CDM08] Javier Civera, Andrew J Davison, and JMM Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[DK06] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[EKC17] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[ESC14] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[EUC16] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, July 2016.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[GLT12] Dorian Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apr 1987.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[KJR+11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, May 2011.

[MAMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[MAT14] Raul Mur-Artal and Juan Tardos. Fast relocalisation and loop closing in keyframe-based SLAM. pages 846–853, 06 2014.

[MSKS03] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[MT16] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[SEE+12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[Vat92] Bala R. Vatti. A generic solution to polygon clipping. *Commun. ACM*, 35(7):56–63, July 1992.