PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments

Ruben Gomez-Ojeda, David Zuñiga-Noël, Francisco-Angel Moreno,
Davide Scaramuzza, and Javier Gonzalez-Jimenez

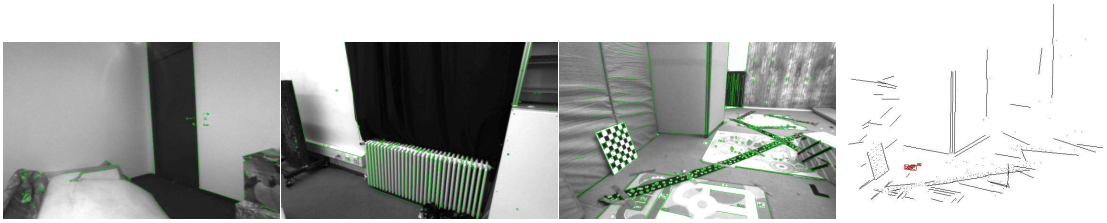
This work has been supported by the Spanish Government (project DPI2017-84827-R and grant BES-2015-071606) and the Andalusian Government (project TEP2012-530). R. Gomez-Ojeda, F.A. Moreno, D. Zuñiga-Noël, and J. Gonzalez-Jimenez are with the Machine Perception and Intelligent Robotics (MAPIR) Group, University of Malaga. (email: rubengooj@gmail.com). D. Scaramuzza is with the Robotics and Perception Group, Dep. of Informatics, University of Zurich, and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland.

Abstract

Traditional approaches to stereo visual SLAM rely on point features to estimate the camera trajectory and build a map of the environment. In low-textured environments, though, it is often difficult to find a sufficient number of reliable point features and, as a consequence, the performance of such algorithms degrades. This paper proposes PL-SLAM, a stereo visual SLAM system that combines both points and line segments to work robustly in a wider variety of scenarios, particularly in those where point features are scarce or not well-distributed in the image. PL-SLAM leverages both points and segments at all the instances of the process: visual odometry, keyframe selection, bundle adjustment, etc. We contribute also with a loop closure procedure through a novel bag-of-words approach that exploits the combined descriptive power of the two kinds of features. Additionally, the resulting map is richer and more diverse in 3D elements, which can be exploited to infer valuable, high-level scene structures like planes, empty spaces, ground plane, etc. (not addressed in this work). Our proposal has been tested with several popular datasets (such as KITTI and EuRoC), and is compared to state of the art methods like ORB-SLAM, revealing a more robust performance in most of the experiments, while still running in real-time. An open source version of the PL-SLAM C++ code will be released for the benefit of the community.

Index Terms: Stereo Visual SLAM, line segment features, bundle adjustment, loop closure

I Introduction



(a) *lt-easy* (b) *euroc/V2-01-easy* (c) *euroc/V1-01-easy* (d) Map from 1(c)

Figure 1: Low-textured environments are challenging for *feature-based* SLAM systems based on traditional keypoints. In contrast, line segments are usually common in human-made environments, and apart from an improved camera localization, the built maps are richer as they are populated with more meaningful elements (3D line-segments).

In recent years, visual Simultaneous Localization And Mapping (SLAM) is firmly progressing towards the degree of reliability required for fully autonomous vehicles: mobile robots, self-driving cars or Unmanned Aerial Vehicles (UAVs). In a nutshell, the SLAM problem consists of the estimation of the vehicle trajectory given as a set of poses (position and orientation), while simultaneously building a map of the environment.

Apart from self-localization, a map becomes useful for obstacle avoidance, object recognition, task planning, etc. [Durrant-Whyte2006a].

As a first-level classification, SLAM systems can be divided into *topological* (e.g. [Milford2004, Cummins2008, Milford2012, Milford2013a]) and *metric* approaches. In this paper, we focus on the latter, which take into account the *geometric* information of the environment and build a physically meaningful map of it [klein09cameraphone, Blanco2015]. These approaches can be further classified into *direct* and *feature-based* systems. The first group, i.e. *direct* methods, estimates the camera motion by minimizing the photometric errors between consecutive frames under the assumption of constant brightness along the local parts of the sequences (examples of this approach can be found elsewhere [newcombe2011dtam, engel2014lsd, forster2016svo]). While this group of techniques has the advantage of working directly with the input images regardless of any intermediate representation, they are very sensitive to brightness changes (this phenomena was addressed in [engel2018direct]) and constrained to narrow baseline motions. In contrast, *feature-based* methods employ an indirect representation of the images, typically in the form of point features, that are tracked along the successive frames and then employed for recovering the pose by minimizing the projection errors [scaramuzza2011visual, murTRO2015].

It is noticeable that the performance of any of the above-mentioned approaches usually decreases in low-textured environments in which it is typically difficult to find a large set of keypoint features. The effect in such cases is an accuracy impoverishment and, occasionally, the complete failure of the system. Many of such low-textured environments, however, contain planar elements that are rich in linear shapes, so it would be possible to extract line segments from them. We claim that these two types of features (keypoints and segments) complement each other and its combination leads to a more versatile, robust and stable SLAM system. Furthermore, the resulting maps comprising both 3D points and segments provide more structural information from the environment than point-only maps, as can be seen in the example shown in Figure 11(d). Thus, applications that perform high-level tasks such as place recognition, semantic mapping or task planning, among others, can significantly benefit from the richer information that can be inferred from them.

These benefits, though, come at the expense of a higher computational burden in both detecting and matching line-segments in images [bartoli2005structure], and also in dealing effectively with segment-specific problems like partial occlusions, line disconnection, etc. which complicate feature tracking and matching as well as the residual computation for the map and pose optimization. Such hurdles are the reason why the number of solutions that have been proposed in the literature to SLAM or Structure from Motion (SfM) with line features (e.g. [mei2006fast, Sola2009, zhang2011hand, hofer2015line3d, briales2016minimal]) is so limited. Besides, the few solutions we have found only perform robustly in highly structured environments while showing unreliable results when applied to more realistic ones such as those recorded in the KITTI or EuRoC datasets. In this work, we address the segment-specific tracking and matching issues by discarding outliers through the comparison of the length and the orientation of the line features, while, for the residual computation, we represent segments in the map with their endpoints coordinates. Thus, the residuals between the observed segments and their corresponding lines in the map are computed by the distance between the projections of those endpoints on the image plane and the infinite lines associated to the observed ones. This way, we are able to build a consistent cost function that seamlessly encompasses both point and line features.

These two kinds of features are also employed to robustly detect loop closures during robot navigation, following a new bag-of-words approach that combines the advantages of using each of them to perform place recognition. In summary, we propose a novel and versatile stereo visual SLAM system, coined PL-SLAM, which builds upon our previous Visual Odometry approach presented in [gomez2016robust], and combines both point and line segment features to perform real-time robot localization and mapping. The main contributions of this work are:

- The first open source stereo SLAM system that employs point and line segment features in real time, hence being capable of operating robustly in low-textured environments where traditional point-only approaches tend to fail, while obtaining similar accuracy in the rest of the scenarios. Because of the consideration of both kinds of features, our proposal also produces rich geometrical maps.
- A new implementation of the bundle adjustment process that seamlessly accounts for both kinds of features while refining the poses of the keyframes.
- An extension of the bag-of-words approach presented in [galvez2012bags] that takes into account the description of both points and line segments to improve the loop-closure process.

A set of illustrative videos showing the performance of proposed system and an open source version of the developed C++ PL-SLAM library are publicly available at http://mapir.uma.es and https://github.com/rubengooj/pl-slam.

II Related Work

Feature-based SLAM is traditionally addressed by tracking keypoints along successive frames and then minimizing some error function (typically based on re-projection errors) to estimate the robot poses [slam16cadena]. Among the most successful proposals we can highlight FastSLAM [montemerlo2002fastslam], PTAM [klein2007parallel] [klein2008improving], SVO [forster2014svo] [forster2016svo], and, more recently, ORB-SLAM [murTRO2015], which relies on a fast and continuous tracking of ORB features [rublee2011orb], and a local bundle adjustment step with the continuous observations of the point features. However, all of the previous approaches tend to fail or reduce their accuracy in low-textured scenarios where the lack of repeatable and reliable features usually hinders the feature tracking process. In the following, we review the state of the art of SLAM systems based on alternative image features to keypoints: i.e. edgelets, lines, or line segments.

One of the remarkable approaches that employs *line* features is the one in [Smith2006], where the authors propose an algorithm to integrate them into a monocular Extended Kalman Filter SLAM system (EKF-SLAM). In the refered paper, the line detection relies on an hypothesize-and-test method that connects several near keypoints to achieve real-time performance. Other works employ *edge* landmarks as features in monocular SLAM, as the one reported in [Eade2009], which does not only include the information of the local planar patch as in the case of keypoints, but also considers local edge segments, hence introducing new valuable information as the orientation of the so-called *edgelets*. In that work they derive suitable models for those kinds of features and use them within a particle-filter SLAM system, achieving nearly real-time performance. More recently, authors in [forster2016svo] also introduced edgelets in combination with intensity corners in order to improve robustness in environments with little or high-frequency texture.

A different approach, known as *model-based*, incorporates prior information about the orientation of the landmarks derived from line segments. Particularly, the method in [zhang2011building] presents a monocular 2D SLAM system that employs vertical and horizontal lines on the floor as features for both motion and map estimation. For that, they propose two different parameterizations for the vertical and the horizontal lines: vertical lines are represented as 2D points on the floor plane (placed the intersection point between the line and such plane), while horizontal lines are represented by their two end-points placed on the floor. Finally, the proposed models is incorporated into an EKF-SLAM system. Another model-based approach is reported in [Zhou2015a], where the authors introduce structural lines in an extension of a standard EKF-SLAM system. The dominant directions of the lines are estimated by computing their vanishing points under the assumption of a Manhattan world [coughlan1999manhattan]. All these model-based approaches, though, are limited to very structured scenarios and/or planar motions, as they rely solely on line features.

The works in [Sola2009, Sola2011] address a generic approach that compares the impact of eight different landmark parametrization for monocular EKF-SLAM, including the use of point and line features. Nevertheless, such systems are only validated through analytic and statistical tools that assumed already known data association and that, unlike our proposal, do not implement a complete front-end that detect and track the line segments. Another technique for building a 3D line-based SLAM system has been proposed in the recent work [zhang2015building]. For that, the authors employ two different representations for the line segments: the Plücker line coordinates for the initialization and 3D projections, and an orthonormal representation for the back-end optimization. Unfortunately, neither the source code is available nor the employed dataset contain any ground-truth, therefore it has not been possible to carry out a comparison against our proposal.

Recently, line segment features have also been employed for monocular pose estimation in combination with points, due to the bad-conditioned nature of this problem. For that, in [gomez2016pl] the authors extended the semi-direct approach in [forster2014svo] with line segments. Thanks to this pipeline, line segments can be propagated efficiently throughout the image sequence, while refining the position of the end-points under the assumptions of high frame rate and very narrow-baseline.

Finally, by the time of the first submission of this paper, a work with the same name (PL-SLAM, [pumarola2017pl]) was published extending the monocular algorithm ORB-SLAM to the case of including line segment features computed through the LSD detector [von2010lsd]. Apart from being a monocular system (unlike our stereo approach), their proposal deals with line tracking and matching in an essentially different way: they propagate the line segments by their endpoints and then perform descriptor-based tracking, which increases the computational burden of ORB-SLAM. Besides this computational drawback, when working with features detected with the LSD detector, the variance of the endpoints becomes quite pronounced, specially in challenging illumination conditions or very low-textured scenes, making more difficult wide-baseline tracking and matching between line features in non-consecutive frames. Our PL-SLAM approach, in contrast, does not make any assumption regarding the position of the lines endpoints so that our tracking front-

end allows to handle partially occluded line segments, endpoints variance, etc., for both the stereo and frame-to-frame tracking, hence becoming a more robust approach to point-and-line SLAM.

III PL-SLAM Overview

The general structure of the PL-SLAM system proposed here is depicted in Figure 2, and its main modules are described in the following sections. As it is common to other SLAM systems (being ORB-SLAM [murTRO2015] the most popular method nowadays), our proposal is also based on three different threads: *visual odometry*, *local mapping*, and *loop closure*. This efficient distribution allows for a continuous tracking of the VO module while the local mapping and the loop closure ones are processed in the background only when a new keyframe is inserted.

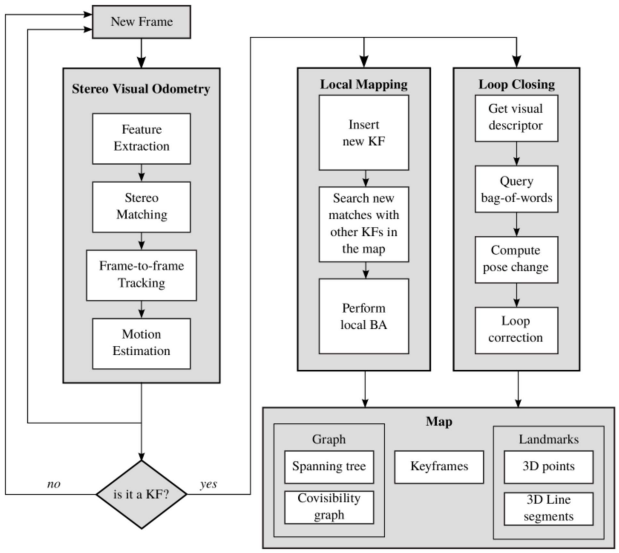


Figure 2: Scheme of the stereo PL-SLAM system.

Map. The map consists of i) a set of keyframes (KFs), ii) the detected 3D landmarks (both keypoints and line segments), iii) a covisibility graph and iv) a spanning tree.

The keyframes contain the observed stereo features and their descriptors, a visual descriptor of the corresponding left image computed through a visual vocabulary as explained later in Section VI-A, and the information of the 3D camera pose.

Regarding the landmarks, we store the list of observations and the most representative descriptor for each detected landmark. Besides, specifically for points, we also keep its estimated 3D position while, for the line segments, we keep both their direction and the estimated 3D coordinates of their endpoints.

Finally, the covisibility information, as in [strasdat2011double], is modeled by a graph: each node represents a KF, and edges between KFs are created only if they share a minimum number of landmarks, which in this work is set to 20 landmarks (see Figure 3 for an example) , allowing for real-time bundle adjustment along the local map.

Similarly, in order to perform a faster loop closure optimization, we also form the so-called *essential graph*, which is less dense than the covisibility graph because an edge between two KFs is created when they share more than 100 landmark observations. Finally, the map also contains a spanning tree, which is the minimum connected representation of a graph that includes all the KFs.

Feature Tracking. We perform feature tracking through the stereo visual odometry algorithm from our previous work [gomez2016robust]. In a nutshell, we track image features (points and segments) from a sequence of stereo frames and compute their 3D position and their associated uncertainty represented by covariance matrices. The 3D landmarks are then projected to the new camera pose, and the projection errors are minimized in order to obtain both the camera pose increment and the covariance associated to such estimation. This process is repeated every new frame, performing simply frame to frame VO, until a new KF is

inserted to the map. Further discussion about this feature tracking procedure will be formally addressed in Section IV. Once a KF is inserted into the map, two procedures are run in parallel: local mapping and loop closure detection.

Local Mapping. The local mapping procedure looks for new feature correspondences between the new KF, the last one and those connected to the last one in the covisibility graph. This way, we build the so-called *local map* of the current KF, which includes all the KFs that share at least 20 landmark observations with the current one as well as all the landmarks observed by them. Finally, an optimization of all the elements within the local map (KF poses and landmarks positions) is performed. A detailed description of this procedure will be presented in Section V.

Loop Closure. In parallel to local mapping, a loop closure detection is carried out by extracting a visual descriptor for each image, based on a bag-of-words approach, as will be described in Section VI. All the visual descriptors of the captured frames during camera motion are stored in a database, which is later employed to find similar frames to the current one. The best match will be considered a loop closure candidate only if the local sequence surrounding this KF is also similar. Finally, the relative $SE(3)$ transformation between the current KF and the loop closure candidate is estimated so that, if a proper estimation is found, all the KFs poses involved in the loop are corrected through a pose-graph optimization (PGO) process.

It is important to remark that the stereo visual odometry system runs continuously at every frame while both the local mapping and loop closure detection procedures are launched in background (in separated threads) only when a new KF is inserted, thus allowing our system to reach real-time performance. In the event of a new keyframe being inserted in the system while the local mapping thread is still being processed, the keyframe is temporary stored until the map is updated and then a new local mapping process is launched.

These mapping and loop closure approaches are identical to the ones followed in ORB-SLAM, being aimed to reduce the high computational burden that general BA involves (along with the incorporation of recent sparse algebra techniques). Within the BA framework, our proposal belongs to the so-called *relative* techniques (e.g. [sibley2009adaptive, sibley2010vast, moreno2016constant]), which have gained great popularity in the last years as an alternative to the more costly *global* approaches (e.g. [klein2007parallel, kaess2008isam]).

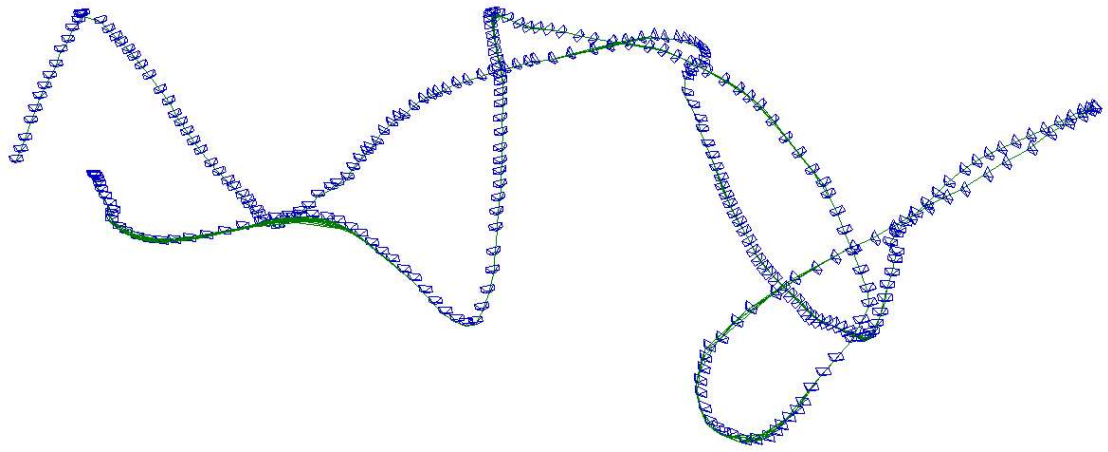


Figure 3: Covisibility graph in the sequence *lt-first* for which we have represented the edges connecting the keyframes with green lines.

IV Feature Tracking

This section reviews the most important aspects of our previous work [gomez2016robust], which deals with the visual odometry estimation between consecutive frames, and also with the KF decision policy. Basically, both points and line segments are tracked along a sequence of stereo frames (see Figure 1), and then the 3D motion of the camera (and also its uncertainty) is computed by minimizing the projection errors.

IV-A *Point Features*

In this work we use the well-known ORB method [rublee2011orb] due to its great performance for keypoint detection, and the binary nature of the descriptor it provides, which allows for a fast, efficient keypoint matching. In order to reduce the number of outliers, we only consider measurements which fulfill that the best match in the left image corresponds to the best match in the right one, i.e. they are mutual best matches. Finally, we also filter out those matches whose distance in the descriptor space with the second best match is less than twice the distance with the best match, to ensure that the correspondences are meaningful enough.

IV-B *Line Segment Features*

The Line Segment Detector (LSD) method [von2010lsd] has been employed to extract line segments, providing high precision and repeatability. For stereo matching and frame-to-frame tracking we augment line segments with a binary descriptor provided by the Line Band Descriptor (LBD) method [zhang2013efficient], which allows us to find correspondences between lines based on their local appearance. Similarly to the case of points, we check that both candidate features are mutual best matches, and also that the feature is meaningful enough. Finally, we take advantage of the useful geometrical information that line segments provide in order to filter out those line matches with different orientations and lengths, and those with a high difference on the disparities of the endpoint. Notice that this filter helps the system to retain a larger amount of structural lines, which allows for the formation of more consistent maps based on points and lines (see Figure 1(d)).

IV-C *Motion Estimation*

Once we have established the correspondences between two stereo frames, we back-project both the keypoints and the line segments from the first frame to the next one. Then, we iteratively estimate the camera ego-motion through a robust Gauss-Newton minimization of the line and keypoint projection errors. In order to deal with outliers, we employ a Pseudo-Huber loss function and perform a two-step minimization, as proposed in [moreno2013erode]. Finally, we obtain the incremental motion estimation between the two consecutive frames, which can be modelled by the following normal distribution:

$$\boldsymbol{\xi}_{t,t+1} \sim \mathcal{N}(\boldsymbol{\xi}^*_{t,t+1}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}^*_{t,t+1}}) \tag{1}$$

where $\boldsymbol{\xi}^*_{t,t+1} \in \mathfrak{se}(3)$ is the 6D vector of the camera motion between the frames t and $t+1$, and $\boldsymbol{\Sigma}_{\boldsymbol{\xi}^*_{t,t+1}}$ stands for the covariance of the estimated motion, approximated by the inverse of the Hessian of the cost function in the last iteration.

IV-D *Keyframe Selection*

For deciding when a new KF is inserted in the map, we have followed the approach in [kerl2013dense] which employs the uncertainty of the relative motion estimation. Thus, following equation (1), we transform the uncertainty from the covariance matrix into a scalar, named *entropy*, through the following expression:

$$h(\boldsymbol{\xi}) = 3(1 + log(2\pi)) + 0.5log(\,|\,\boldsymbol{\Sigma}_{\boldsymbol{\xi}}\,|\,) \tag{2}$$

Then, for a given KF i we check the ratio between the entropy from the motion estimation between the previous KF i and the current one $i+u$ and that between the previous KF i and its first consecutive frame $i+1$, i.e.:

$$\alpha = \frac{h(\boldsymbol{\xi}_{i,i+u})}{h(\boldsymbol{\xi}_{i,i+1})} \tag{3}$$

If the value of α lies below some pre-established threshold, which in our experiments has been set to 0.9, then the frame $i+u$ is inserted to the system as a new KF. Notice that to compute the expression in Equation (2), we need the uncertainty of the pose increment between non-consecutive frames. Since Equation (1) only estimates the incremental motion between consecutive frames, a series of such estimations are composed through first order Gaussian propagation techniques to obtain the covariance between two non-consecutive KFs.

V Local Mapping

This section describes the behavior of the system when a new KF is inserted, which essentially consists in performing the bundle adjustment of the so-called *local map* i.e.: those KFs connected with the current one by the covisibility graph and the landmarks observed by those local KFs.

V-A *Keyframe Insertion*

Every time the visual odometry thread selects a KF, we insert it into the SLAM system and optimize the local map. First, we refine the estimation of the relative pose change between the current and the previous KFs, since the one provided by the VO is estimated by composing the relative motions between the intermediate frames. For that, we perform data association between the KFs, taking into account the geometrical restrictions described in Section IV and obtaining a consistent set of common features observed in them. Then, we perform a similar optimization than the one presented in Section IV-C, for which we employ the pose provided by the VO thread as the initial estimation for a Gauss-Newton minimization. Once we have computed the relative pose change between the KFs, we insert the current one into the system, including:

1. An index for the keyframe.
2. The information of its 3D pose, which comprises an absolute pose and the relative pose from the previous KF, along with their associated uncertainties.
3. The new 3D landmarks, which are initialized by storing both their 2D image coordinates and their descriptors. The new observations of the already existing landmarks are also added to the map.

Finally, we also look for new correspondences between the unmatched feature observations from the current frame, and the landmarks in the local map.

V-B *Local Bundle Adjustment*

After inserting the KF, the next step is to perform a bundle adjustment of the local map. As stated before, this map is formed by all the KFs connected with the current one in the covisibility graph (i.e. those that share at least 20 landmarks) and also all the landmarks observed by the local KFs. For that, let us define the vector $\boldsymbol{\psi}$ that contains the variables to be optimized, which are the $\mathfrak{se}(3)$ pose of each KF $\boldsymbol{\xi}_{iw}$, the 3D position of each point $\mathbf{X}_{wj}$, and also the 3D positions of the endpoints for each line segment: $\{\mathbf{P}_{wk}, \mathbf{Q}_{wk}\}$. Then, we minimize the projection errors between the observations and the landmarks projected to the frames where they were observed:

$$\boldsymbol{\psi}^* = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{i \in \mathcal{K}_l} \left[\sum_{j \in \mathcal{P}_l} \mathbf{e}_{ij}{}^\top \boldsymbol{\Sigma}_{\mathbf{e}_{ij}}{}^{-1} \mathbf{e}_{ij} + \sum_{k \in \mathcal{L}_l} \mathbf{e}_{ik}{}^\top \boldsymbol{\Sigma}_{\mathbf{e}_{ik}}{}^{-1} \mathbf{e}_{ik} \right] \tag{4}$$

where $\mathcal{K}_l$, $\mathcal{P}_l$ and $\mathcal{L}_l$ refer to the groups of local KFs, points, and line segments, respectively.

In this expression, the projection error $\mathbf{e}_{ij}$ stands for the 2D distance between the observation of the j-th map point in the i-th KF, and can be expressed as:

$$\mathbf{e}_{ij} = \mathbf{x}_{ij} - \boldsymbol{\pi}(\boldsymbol{\xi}_{iw}, \mathbf{X}_{wj}) \tag{5}$$

where the function $\boldsymbol{\pi} : \mathfrak{se}(3) \times \mathbb{R}^3 \mapsto \mathbb{R}^2$ first places the j-th 3D point $\mathbf{X}_{wj}$ (in world coordinates) into the local reference system of the i-th KF, i.e. $\mathbf{X}_{ij}$, and then projects this point to the image. The use of line segments is slightly different, since we cannot simply compare the position of the endpoints as they might be displaced along the line or occluded from one frame to the next one. For that, we take as error function the distances between the projected endpoints of the 3D line segment and its corresponding infinite line in the image plane. In this case, the error $\mathbf{e}_{ik}$ between the k-th line observed in the i-th frame, is given by:

$$\mathbf{e}_{ik} = \begin{bmatrix} \mathbf{l}_{ik} \cdot \boldsymbol{\pi}(\boldsymbol{\xi}_{iw}, \mathbf{P}_{wk}) \\ \mathbf{l}_{ik} \cdot \boldsymbol{\pi}(\boldsymbol{\xi}_{iw}, \mathbf{Q}_{wk}) \end{bmatrix} \tag{6}$$

where $\mathbf{P}_{wk}$ and $\mathbf{Q}_{wk}$ refer to the 3D endpoints of the line segments in the world coordinate system and $\mathbf{l}_{ik}$ is the equation of the infinite line that corresponds to the k-th line segment in the i-th KF, which can be obtained with the cross product between the 2D endpoints of the line segments in homogeneous coordinates, i.e.: $\mathbf{l}_{ik} = \mathbf{p}_{ik} \times \mathbf{q}_{ik}$.

The problem in (4) can be iteratively solved by following the Levenberg-Marquardt optimization approach, for which we need to estimate both the Jacobian and the Hessian matrices:

$$\Delta\boldsymbol{\psi} = \left[\mathbf{H} + \lambda diag(\mathbf{H}) \right]^{-1} \mathbf{J}^\top \mathbf{W} \mathbf{e} \tag{7}$$

where the error vector $\mathbf{e}$ contains all the projection errors $\mathbf{e}_{ij}$ and $\mathbf{e}_{ik}$. This equation, along with the following update step:

$$\boldsymbol{\psi}' = \boldsymbol{\psi} \boxplus \Delta\boldsymbol{\psi} \tag{8}$$

$$\mathbf{H} \approx \left[\& \vdots \frac{\partial \mathbf{e}_{ij}}{\partial \boldsymbol{\xi}_{iw}}^{\top} \overset{\cdots}{\underset{\cdots}{\boldsymbol{\Sigma}_{\mathbf{e}_{ij}}^{-1}}} \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{X}_{wj}} \vdots\vdots \frac{\partial \mathbf{e}_{ik}}{\partial \boldsymbol{\xi}_{iw}}^{\top} \overset{\cdots}{\underset{\cdots}{\boldsymbol{\Sigma}_{\mathbf{e}_{ik}}^{-1}}} \frac{\partial \mathbf{e}_{ik}}{\partial \{\mathbf{P}_{wk}, \mathbf{Q}_{wk}\}} \vdots\vdots \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{X}_{wj}}^{\top} \overset{\cdots}{\underset{\cdots}{\boldsymbol{\Sigma}_{\mathbf{e}_{ij}}^{-1}}} \frac{\partial \mathbf{e}_{ij}}{\partial \boldsymbol{\xi}_{iw}} \vdots\vdots \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{X}_{wj}}^{\top} \overset{\cdots}{\underset{\cdots}{\boldsymbol{\Sigma}_{\mathbf{e}_{ij}}^{-1}}} \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{X}_{wj}} \vdots\vdots \overset{\cdots}{\underset{\cdots}{\mathbf{0}}} \vdots\vdots \frac{\partial \mathbf{e}_{ik}}{\partial \{\mathbf{P}_{wk}, \mathbf{Q}_{wk}\}}^{\top} \overset{\cdots}{\underset{\cdots}{\boldsymbol{\Sigma}_{\mathbf{e}_{ik}}^{-1}}} \frac{\partial \mathbf{e}_{ik}}{\partial \boldsymbol{\xi}_{iw}} \vdots$$

Conversion to HTML had a Fatal error and exited abruptly. This document may be truncated or damaged.