PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines

Albert Pumarola[1] Alexander Vakhitov[2] Antonio Agudo[1] Alberto Sanfeliu[1] Francesc Moreno-Noguer[1]

***Abstract*— Low textured scenes are well known to be one of the main Achilles heels of geometric computer vision algorithms relying on point correspondences, and in particular for visual SLAM. Yet, there are many environments in which, despite being low textured, one can still reliably estimate line-based geometric primitives, for instance in city and indoor scenes, or in the so-called "Manhattan worlds", where structured edges are predominant. In this paper we propose a solution to handle these situations. Specifically, we build upon ORB-SLAM, presumably the current state-of-the-art solution both in terms of accuracy as efficiency, and extend its formulation to simultaneously handle both point and line correspondences. We propose a solution that can even work when most of the points are vanished out from the input images, and, interestingly it can be initialized from solely the detection of line correspondences in three consecutive frames. We thoroughly evaluate our approach and the new initialization strategy on the TUM RGB-D benchmark and demonstrate that the use of lines does not only improve the performance of the original ORB-SLAM solution in poorly textured frames, but also systematically improves it in sequence frames combining points and lines, without compromising the efficiency.**

I. INTRODUCTION

The last years have witnessed a surge in autonomous cars and aerial vehicles able to navigate for hundreds of miles without human intervention [10], [16], [32]. Among other technologies, at the core of these systems lie sophisticated Simultaneous Localization And Mapping (SLAM) algorithms, which have proven effective to accurately estimate trajectories while geometrically reconstructing the unknown environment.

Since the groundbreaking Parallel Tracking And Mapping (PTAM) [13] algorithm was introduced by Klein and Murray in 2007, many other real-time visual SLAM approaches have been proposed, including the feature point-based ORB-SLAM [18], and the direct-based methods LSD-SLAM [7] and RGBD-SLAM [6] that optimize directly over image pixels. Among them, the ORB-SLAM [18] seems to be the current state-of-the-art, yielding better accuracy than the direct methods counterparts.

While the performance of ORB-SLAM [18] in well textured sequences is impressive, it is prone to fail when dealing with poorly textured videos or when feature points are temporary vanished out due to, *e.g.*, motion blur. This kind of situations are often encountered in man-made scenarios. However, despite the lack of reliable feature points, these environments may still contain a number of lines that can be used in a similar way.

[1]A.Pumarola, A.Agudo, A.Sanfeliu and F.Moreno-Noguer are with the Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Barcelona, Spain

[2]A.Vakhitov is with Skolkovo Institute of Science and Technology, Moscow, Russia.

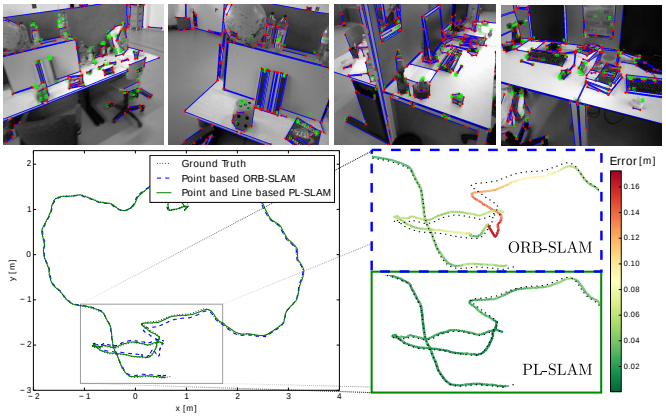

Fig. 1. ORB-SLAM [18] vs PL-SLAM. **Top**: The proposed PL-SLAM allows to simultaneously handle point and line features. This is specially advantageous in situations with small number of points such as that shown in the second image. **Bottom-Left**: Comparison of the trajectories obtained using the state-of-the-art point-based method ORB-SLAM [18] and our PL-SLAM, in a TUM RGB-D sequence. The black dotted line shows the ground truth, the blue dashed line is the trajectory obtained with ORB-SLAM [18], and the green solid line is the trajectory obtained with PL-SLAM. **Bottom-Right**: Close-up of part of the map color-coded with the amount of error. Red corresponds to higher error levels, and green to lower ones. Note how the use of lines consistently improves the accuracy of the estimated trajectory.

Exploiting lines, though, is not a trivial task. First, existing line detectors and parameterizations are not as well-established in the literature as feature point ones. And secondly, the algorithms to compute pose from line correspondences are less reliable than those based on points and are very sensitive to the partial occlusions that lines may undergo. These reasons made that current SLAM approaches making use of lines rely on range cameras or laser scanners [2], [12], [20], [25].

In this work, we tackle all these issues using a *purely* visual-based approach. Building upon the ORB-SLAM [18] framework, we propose PL-SLAM (Point and Line SLAM), a solution that can simultaneously leverage points and lines information. As recently suggested by [30], lines are parameterized by their endpoints, whose exact location in the image plane is estimated following a two-step optimization process. This representation, besides yielding robustness to occlusions and mis-detections, allows integrating the line representation within the SLAM machinery as if they were points and hence re-use most of the ORB-SLAM [18] architecture. The resulting approach is shown to be very accurate in poorly textured environments, and also, improves the performance of the original ORB-SLAM [18] in highly textured sequences (see Fig. 1).

An additional contribution of this paper is that we also propose a new initialization approach that allows estimating an approximate initial maps from only line correspondences between three consecutive images. Previous solutions were based on homography [8] or essential matrix estimation [29], and required point correspondences. To the best of our knowledge, there are no equivalent techniques based on lines. The solution we propose holds on the assumption of constant rotation between three consecutive frames and that these rotations are relatively small. In the experimental section, we will show that despite these approximations, the initial map we estimate highly resembles those obtained by point-based solutions, and therefore, are a very good alternative to use when feature points are not available.

II. RELATED WORK

Building the 3D rigid structure of unknown environment while recovering the camera trajectory from a monocular image sequence has been an extremely important research area in robotics and computer vision for decades, with many real applications in autonomous robot navigation and augmented reality. This problem is known as SLAM, and its core is roughly the same compared to structure-from-motion algorithms.

Early filtering approaches applied the Extended Kalman Filter (EKF) [5] to process every frame in the video for small maps, providing the first real-time solutions. Subsequent works based on Bundle Adjustment (BA) handled denser maps just using key-frames to estimate the map [13], [17], obtaining more accurate solutions [27] than filtering techniques. Most approaches rely on PTAM algorithm [13], that represented a breakthrough in visual-based SLAM. This method approximately decouples localization and mapping in two threads that run in parallel, relying on FAST corners points [23]. In [14] the accuracy was improved with edge features together with a rotation estimation step during tracking that provided better relocalization results, and even reducing the computational cost [24]. More recently, the ORB-SLAM system has been proposed in [18], providing a more robust camera tracking and mapping estimator. A multi-threaded CPU approach was presented in [7] to estimate real-time dense structure estimation.

However, all previous feature-based methods fail in environments with poor texture or situations with defocus and motion blur. To solve this, dense and direct methods can be applied, even though they are likely to be computationally expensive [19], [21], and require dedicated GPU-implementations to achieve real-time performance. Other semi-direct methods such as [9] overcome the high-computation requirement of dense methods by exploiting only pixels with strong gradients, providing an intermediate level of accuracy, density and complexity. Scene prior information have been also exploited to provide a significant boost to SLAM systems [3], [4].

Motivated by the need for efficient and accurate scene representations even for poorly textured environments, in tasks such as visual inspection from aerial vehicles or hand-held devices (i.e., with limited computational resources), we here propose a novel visual-based SLAM system that can combine points and lines information in a unified framework while keeping the computational cost. Note that several parametrizations to combine points and lines were used in EKF-SLAM [26]. However, as we said above, filtering-based approaches have been outperformed by optimization-based approaches in rigid SLAM, as we do in this work. We validate our method on a wide variety of scenarios, outperforming state-of-the-art solutions for highly textured sequences and showing very accurate solutions in low-textured scenarios where standard feature-based methods fail.

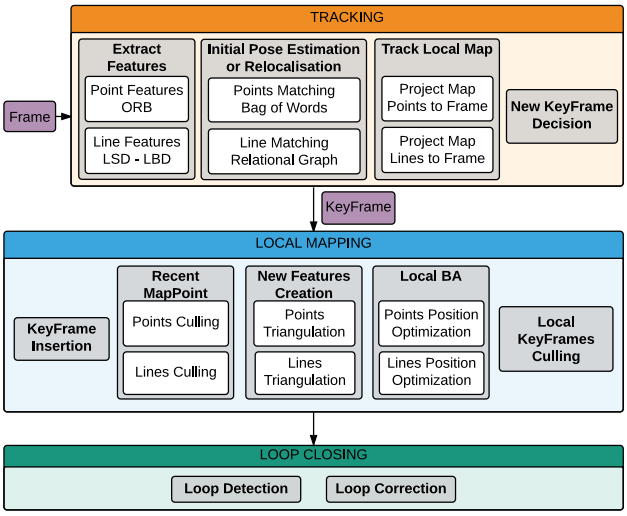

Fig. 2. PL-SLAM pipeline, an extension of the ORB-SLAM [18] pipeline. The system is composed by three main threads: *Tracking*, *Local Mapping* and *Loop Closing*. The *Tracking* thread estimates the camera position and decides when to add new keyframes. Then, *Local Mapping* adds the new keyframe information into the map and optimizes it with BA. The *Loop Closing* thread is constantly checking for loops and correcting them.

III. SYSTEM OVERVIEW

The pipeline of our approach highly resembles that of the ORB-SLAM [18], in which we have integrated the information provided by line features (see Fig. 2). We next briefly review the main building blocks in which line operations are performed. For a description of the operations involving point features, the reader is referred to [18].

One of the main issues to address in SLAM algorithms is the computational complexity. In order to preserve the real-time characteristics of ORB-SLAM [18], we have carefully chosen, used and implemented fast methods for operating with lines in all stages of the pipeline: detection, triangulation, matching, culling, relocalization and optimization. Line segments in an input frame are detected by mean of LSD [31], an $O(n)$ line segment detector, where n is the number of pixels in the image. Then, lines are pairwise matched with lines already present in the map using a relational graph strategy [33]. This approach relies on lines' local appearance (Line Band Descriptors) and geometric

constraints and is shown to be quite robust against image artifacts while preserving the computational efficiency.

As it is done with point features, after having obtained an initial set of map-to-image line feature pairs, all lines of the local map are projected onto the image to find further correspondences. Then, if the image contains sufficient new information about the environment, it is flagged as a keyframe and its corresponding lines are triangulated and added to the map. To discard possible outliers, lines seen from less than three viewpoints or in less than 25% of the frames from which they were expected to be seen are discarded too (culling). Line positions in the map are optimized with a local BA. Note in Fig. 2 that we do not use lines for loop closing. Matching lines across the whole map is too computationally expensive. Hence, only point features are used for loop detection.

IV. LINE-BASED SLAM

We next describe the line parameterization and error function we use and how this is integrated within the main building blocks of the SLAM pipeline, namely bundle adjustment, global relocalization and feature matching.

A. Line-based Reprojection Error

In order to extend the ORB-SLAM [18] to lines, we need a proper definition of the reprojection error and line parameterization.

Following [30], let $\mathbf{P},\mathbf{Q} \in \mathbb{R}^3$ be the 3D endpoints of a line, $\mathbf{p}_\mathrm{d},\mathbf{q}_\mathrm{d} \in \mathbb{R}^2$ their 2D detections in the image plane, and $\mathbf{p}^\mathrm{h}_\mathrm{d},\mathbf{q}^\mathrm{h}_\mathrm{d} \in \mathbb{R}^3$ theirs corresponding homogeneous coordinates. From the latter we can obtain the normalized line coefficients as:

$$\mathbf{l} = \frac{\mathbf{p}^\mathrm{h}_\mathrm{d} \times \mathbf{q}^\mathrm{h}_\mathrm{d}}{\left|\mathbf{p}^\mathrm{h}_\mathrm{d} \times \mathbf{q}^\mathrm{h}_\mathrm{d}\right|}. \tag{1}$$

The *line reprojection error* E_line is then defined as the sum of point-to-line distances E_pl between the projected line segment endpoints, and the detected line in the image plane (see Fig. 3-right). That is:

$$\mathrm{E}_\mathrm{line}(\mathbf{P},\mathbf{Q},\mathbf{l},\theta,\mathbf{K}) = \mathrm{E}^2_\mathrm{pl}(\mathbf{P},\mathbf{l},\theta,\mathbf{K}) + \mathrm{E}^2_\mathrm{pl}(\mathbf{Q},\mathbf{l},\theta,\mathbf{K}), \tag{2}$$

with:

$$\mathrm{E}_\mathrm{pl}(\mathbf{P},\mathbf{l},\theta,\mathbf{K}) = \mathbf{l}^\top \pi(\mathbf{P},\theta,\mathbf{K}), \tag{3}$$

where $\mathbf{l}$ are the detected line coefficients, $\pi(\mathbf{P},\theta,\mathbf{K})$ represents the projection of the endpoint $\mathbf{P}$ onto the image plane, given the internal camera calibration matrix $\mathbf{K}$, and the camera parameters $\theta = \{\mathbf{R},\mathbf{t}\}$ that includes the rotation and translation parameters, respectively.

Note that in practice, due to real conditions such as line occlusions or mis-detections, the image detected endpoints $\mathbf{p}_\mathrm{d}$ and $\mathbf{q}_\mathrm{d}$ will not match the projections of the endpoints $\mathbf{P}$ and $\mathbf{Q}$ (see Fig. 3-left). Therefore, we define the *detected line reprojection error* as:

$$\mathrm{E}_\mathrm{line,d}(\mathbf{p}_\mathrm{d},\mathbf{q}_\mathrm{d},\mathbf{l}) = \mathrm{E}^2_\mathrm{pl,d}(\mathbf{p}_\mathrm{d},\mathbf{l}) + \mathrm{E}^2_\mathrm{pl,d}(\mathbf{q}_\mathrm{d},\mathbf{l}), \tag{4}$$

where $\mathbf{l}$ is the projected 3D line coefficients and the detected point-to-line error is $\mathrm{E}_\mathrm{pl,d}(\mathbf{p}_\mathrm{d},\mathbf{l}) = \mathbf{l}^\top \mathbf{p}_\mathrm{d}$.

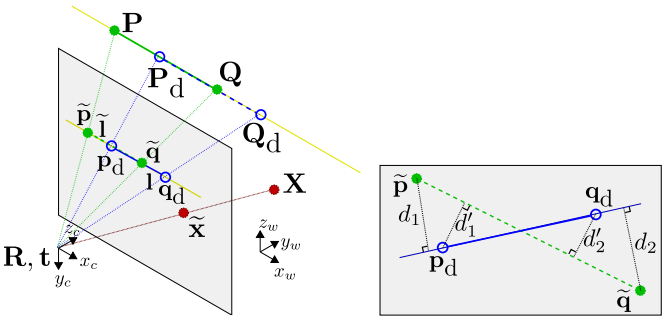

Fig. 3. **Left**: Notation. Let $\mathbf{P},\mathbf{Q} \in \mathbb{R}^3$ be the 3D endpoints of a 3D line, $\tilde{\mathbf{p}},\tilde{\mathbf{q}} \in \mathbb{R}^2$ their projected 2D endpoints to the image plane and $\tilde{\mathbf{l}}$ the projected line coefficients. $\mathbf{p}_\mathrm{d},\mathbf{q}_\mathrm{d} \in \mathbb{R}^2$ the 2D endpoints of a detected line, $\mathbf{P}_\mathrm{d},\mathbf{Q}_\mathrm{d} \in \mathbb{R}^3$ their real 3D endpoints, and $\mathbf{l}$ the detected line coefficients. $\mathbf{X} \in \mathbb{R}^3$ is a 3D point and $\tilde{\mathbf{x}} \in \mathbb{R}^2$ its corresponding 2D projection. **Right**: Line-based reprojection error. d_1 and d_2 represent the *line reprojection error*, and d'_1 and d'_2 the *detected line reprojection error* between a detected 2D line (blue solid) and the corresponding projected 3D line (green dashed).

Based on the methodology proposed in [30], a recursion over the detected reprojection line error will be applied in order to optimize the pose parameters θ while approximating $\mathrm{E}_\mathrm{line,d}$ to the line error E_line defined on Eq. (2).

B. Bundle Adjustment with Points and Lines

The camera pose parameters $\theta = \{\mathbf{R},\mathbf{t}\}$ are optimized at each frame with a BA strategy that constrains θ to lie in the SE(3) group. For doing this, we build upon the framework of the ORB-SLAM [18] but besides feature point observations, we include the lines as defined in the previous subsection. We next define the specific cost function we propose to be optimized by the BA that combines the two types of geometric entities.

Let $\mathbf{X}_j \in \mathbb{R}^3$ be the generic j-th point of the map. For the i-th keyframe, this point can be projected onto the image plane as:

$$\tilde{\mathbf{x}}_{i,j} = \pi(\mathbf{X}_j,\theta_i,\mathbf{K}), \tag{5}$$

where $\theta_i = \{\mathbf{R}_i,\mathbf{t}_i\}$ denotes the specific pose of the i-th keyframe. Given an observation $\mathbf{x}_{i,j}$ of this point, we define following 3D error:

$$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \tilde{\mathbf{x}}_{i,j}\ . \tag{6}$$

Similarly, let us denote by $\mathbf{P}_j$ and $\mathbf{Q}_j$ the endpoints of the j-th map line segment. The corresponding image projections (expressed in homogeneous coordinates) onto the same keyframe can be written as:

$$\tilde{\mathbf{p}}^\mathrm{h}_{i,j} = \pi(\mathbf{P}_j,\theta_i,\mathbf{K}), \tag{7}$$

$$\tilde{\mathbf{q}}^\mathrm{h}_{i,j} = \pi(\mathbf{Q}_j,\theta_i,\mathbf{K})\ . \tag{8}$$

Then, given the image observations $\mathbf{p}_{i,j}$ and $\mathbf{q}_{i,j}$ of the j-th line endpoints, we use Eq. (1) to estimate the coefficients of the observed line $\tilde{\mathbf{l}}_{i,j}$. We define the following error vectors for the line:

$$\mathbf{e}'_{i,j} = (\tilde{\mathbf{l}}_{i,j})^\top (\mathbf{K}^{-1}\mathbf{p}^\mathrm{h}_{i,j}), \tag{9}$$

$$\mathbf{e}''_{i,j} = (\tilde{\mathbf{l}}_{i,j})^\top (\mathbf{K}^{-1}\mathbf{q}^\mathrm{h}_{i,j}). \tag{10}$$

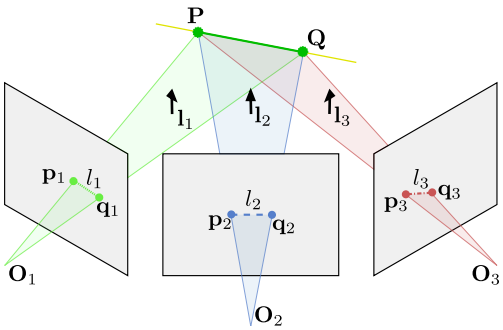

Fig. 4. Estimating camera rotation from line correspondences. $\mathbf{P},\mathbf{Q} \in \mathbb{R}^3$ are the 3D line endpoints, l_i, $i = \{1,2,3\}$ its detections in three consecutive frames with endpoints $\mathbf{p}_i, \mathbf{q}_i$, and coefficients $\mathbf{l}_i$.

The errors (9, 10) are in fact instances of the point-to-line error (3). As explained in [30] they are not constant w.r.t. shift of the endpoints $\mathbf{P}_j$, $\mathbf{Q}_j$ along the corresponding 3D line, which serves as implicit regularization allowing us to use such a non-minimal line parametrization in the BA.

Observe that representing lines using their endpoints we obtain comparable error representations for points and lines. We can therefore build a unified cost function that integrates each of the error terms as:

$$C = \sum_{i,j} \rho \left(\mathbf{e}_{i,j}^{\top} \Omega_{i,j}^{-1} \mathbf{e}_{i,j} + \mathbf{e}_{i,j}^{'\top} \Omega_{i,j}^{'-1} \mathbf{e}_{i,j}^{'} + \mathbf{e}_{i,j}^{''\top} \Omega_{i,j}^{''-1} \mathbf{e}_{i,j}^{''} \right)$$

where ρ is the Huber robust cost function and $\Omega_{i,j}$, $\Omega'_{i,j}$, $\Omega''_{i,j}$ are the covariance matrices associated to the scale at which the keypoints and line endpoints were detected, respectively.

C. Global Relocalization

An important component of any SLAM method, is an approach to relocalize the camera when the tracker is lost. This is typically achieved by means of a PnP algorithm, that estimates the pose of the current (lost) frame given correspondences with 3D map points appearing in previous keyframes. On top of the PnP method, a RANSAC strategy is used to reject outliers correspondences.

In the ORB-SLAM [18], the specific PnP method that is used is the EPnP [1], which however, only accepts point correspondences as inputs. In order to make our approach appropriate to handle lines for relocalization, we have replaced the EPnP by the recently published EPnPL [30], which minimizes the *detected line reprojection error* of Eq. (4).

Furthermore, EPnPL [30] is robust to partial line occlusion and mis-detections. This is achieved by means of a two-step procedure in which first minimizes the reprojection error of the detected lines and estimates the line endpoints $\mathbf{p}_\mathrm{d}, \mathbf{q}_\mathrm{d}$. These points, are then shifted along the line in order to match the projections $\tilde{\mathbf{p}}_\mathrm{d}, \tilde{\mathbf{q}}_\mathrm{d}$ of the 3D model endpoints $\mathbf{P},\mathbf{Q}$ (see Fig. 3). Once these matches are established, the camera pose can be reliably estimated.

V. MAP INITIALIZATION WITH LINES

Another contribution of this paper is an algorithm to estimate an initial map using only line correspondences. Current optimization-based SLAM approaches are initialized with maps built from point correspondences between at least two frames. Homography [8] or essential matrix [29] estimation algorithms are then used to compute the initial map and pose parameters. We next describe our line-based solution for map initialization, which can be a good alternative in low textured scenes with lack of feature points.

Let us consider the setup of Fig. 4, where a line defined by endpoints $\mathbf{P},\mathbf{Q}$ is projected onto three camera views. Let $\{\mathbf{p}_1,\mathbf{q}_1\}$, $\{\mathbf{p}_2,\mathbf{q}_2\}$ and $\{\mathbf{p}_3,\mathbf{q}_3\}$ be the endpoint projections in each of the views and $\mathbf{l}_1,\mathbf{l}_2,\mathbf{l}_3 \in \mathbb{R}^3$ the corresponding line coefficients computed from the projected endpoints.

We will make the assumption of small and continuous rotation between consecutive camera poses, such that the rotation from the first to the second camera views is the same than the rotation from the second to the third one[1]. Under this assumption we can represent the three camera rotations by $\mathbf{R}_1 = \mathbf{R}^{\top}$, $\mathbf{R}_2 = \mathbf{I}$, and $\mathbf{R}_3 = \mathbf{R}$, with $\mathbf{I}$ being the 3×3 identity matrix.

Note that the line coefficients $\mathbf{l}_i$, $i = \{1,2,3\}$ also represent the parameters of a vector which is normal to the plane formed by the center of projection $\mathbf{O}_i$ and the projections $\mathbf{p}_i, \mathbf{q}_i$. The cross product of two such vectors $\mathbf{l}_i$ will be parallel to the line $\mathbf{P},\mathbf{Q}$ and at the same time orthogonal to the third vector, all of them appropriately rotated and put in a common reference. This constraint can be written as:

$$\mathbf{l}_2^{\top} \left((\mathbf{R}^{\top}\mathbf{l}_1) \times (\mathbf{R}\mathbf{l}_3) \right) = 0. \tag{11}$$

Additionally, for small rotations we can approximate $\mathbf{R}$ as:

$$\mathbf{R} = \begin{pmatrix} 1 & -r_3 & r_2 \\ r_3 & 1 & -r_1 \\ -r_2 & r_1 & 1 \end{pmatrix}. \tag{12}$$

For this parametrization, having three matched lines, we will have three quadratic equations like Eq. (11) with three unknowns, r_1, r_2 and r_3. We adapt the polynomial solver of [15], which yields up to eight solutions. For each possible rotation matrix we can get $\mathbf{t}_1,\mathbf{t}_3$ by using the trifocal tensor equations [11] which will be linear in $\mathbf{t}_1,\mathbf{t}_3$. We assume $\mathbf{t}_2 = \mathbf{0}$. We evaluate the eight possible solutions and keep the one that minimizes Eq. (11).

It is worth to point that in order to get enough independent constraints when solving for the translation components using the trifocal tensor equations, we need two additional line correspondences, and hence, the total number of line matches required by our algorithm is five.

VI. EXPERIMENTAL RESULTS

We have compared our system with the current state-of-the-art Visual SLAM methods using the TUM RGB-D benchmark [28]. Also, we evaluate the proposed initialization approach with synthetic and real data and compare the computation time of our PL-SLAM algorithm and the ORB-SLAM [18]. All experiments were carried out with an Intel

[1]In the experimental section we will evaluate the consequences of this assumption, and show that in practice is a good approximation.

TABLE I

LOCALIZATION ACCURACY IN THE TUM RGB-D BENCHMARK [28]

Absolute KeyFrame Trajectory RMSE [cm]						
TUM RGB-D Sequence	PL-SLAM Classic Init	PL-SLAM Line Init	ORB-SLAM	PTAM†	LSD-SLAM†	RGBD-SLAM†
f1_xyz	1.21	1.46	1.38	**1.15**	9.00	1.34
f2_xyz	0.43	1.49	0.54	**0.2**	2.15	2.61
f1_floor	7.59	9.42	8.71	-	38.07	**3.51**
f2_360_kidnap	3.92	60.11	4.99	**2.63**	-	393.3
f3_long_office	**1.97**	5.33	4.05	-	38.53	-
f3_nstr_tex_far	ambiguity detected	37.60	ambiguity detected	34.74	**18.31**	-
f3_nstr_tex_near	2.06	**1.58**	2.88	2.74	7.54	-
f3_str_tex_far	**0.89**	1.25	0.98	0.93	7.95	-
f3_str_tex_near	1.25	7.47	1.5451	**1.04**	-	-
f2_desk_person	**1.99**	6.34	5.95	-	31.73	6.97
f3_sit_xyz	**0.066**	9.03	0.08	0.83	7.73	-
f3_sit_halfsph	**1.31**	9.05	1.48	-	5.87	-
f3_walk_xyz	**1.54**	ambiguity detected	1.64	-	12.44	-
f3_walk_halfsph	**1.60**	ambiguity detected	2.09	-	-	-

Median over 5 executions for each sequence. All trajectories were aligned with 7DoF with the ground truth before computing the ATE error with the script provided by the benchmark [28]. Both ORB-SLAM and PL-SLAM were executed with the parametrization of the on-line open source ORB-SLAM package. †Result of PTAM, LSD-SLAM and RGBD-SLAM were extracted from [18].

TABLE II

TRACKING AND MAPPING TIMES

Mean execution time [ms]			
Thread	Operation	PL-SLAM	ORB-SLAM
Local Mapping	KeyFrame Insertion	17.08	9.86
	Map Feature Culling	1.18	1
	Map Features Creation	74.64	8.39
	Local BA	218.25	118.5
	KeyFrame Culling	12.7	2.86
	Total	3Hz	7Hz
Tracking	Features Extraction	31.32	10.76
	Initial Pose Estimation	7.16	7.16
	Track Local Map	12.58	3.18
	Total	20Hz	50Hz

Mean execution time of 5 different sequences of the TUM RGB-D benchmark [28].

Core i7-4790 (4 cores @3.6 GHz), 8Gb RAM and ROS Hydro [22]. Due to the randomness of the some stages of the pipeline, e.g., initialization, position optimization or global relocalization, all experiments were run five times and we report the median of all executions. Supplementary material can be found on website `http://www.albertpumarola.com/research/pl-slam/`.

A. Localization Accuracy in the TUM RGB-D Benchmark

To evaluate the localization accuracy we compare our PL-SLAM method against current state-of-the-art Visual SLAM methods, including ORB-SLAM [18], PTAM [13], LSD-SLAM [7] and RGBD-SLAM [6]. The metric used for the comparison is the Absolute Trajectory Error (ATE), provided by the evaluation script of the benchmark. Before computing the error, all trajectories are aligned using a similarity warp except for the RGBD-SLAM [6] which is aligned by a rigid body transformation. The results are summarized in Table I.

Note that our PL-SLAM consistently improves the trajectory accuracy of ORB-SLAM [18] in all sequences. Indeed, it yields the best result in all but two sequences, for which PTAM [13] performs slightly better. Nevertheless, PTAM [13] turned not to be so reliable, as in 5 out of all 12 sequences it lost track. LSD-SLAM [7] and RGBD-SLAM [6] also lost track in 3 and 7 sequences, respectively.

B. Map Initialization - Synthetic Experiments

In order to evaluate the map initialization algorithm we describe in Sect. V we perform several synthetic and real experiments.

In the synthetic tests we first evaluate the stability of the polynomial solver we built, modifying the toolbox of Kukelova *et al.* [15]. Fig. 5-left shows the distribution of errors in the parameter estimation for ideal solutions. Note that the average error is around 1e-15, indicating that our modified solver is very stable.

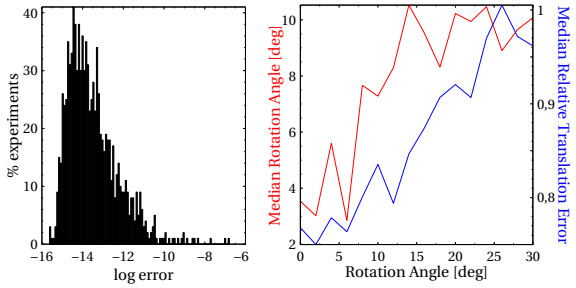

Fig. 5. Map Initialization - Synthetic experiments. **Left**: Numerical stability of the polynomial system solver. **Right**: Rotation and translation error w.r.t frames rotation.

Additionally, we have assessed the consequences of assuming small and constant rotations between three consecutive frames. Fig. 5-right displays the rotation and translation errors produced for increasing inter-frame rotations. While the estimated rotation error remains within relatively small bounds, the translation error is more severely affected by the small rotation assumption. In any event, when this initial map is fed into the BA optimizer, the translation error is drastically reduced.

C. Map Initialization - Real Experiments

We also evaluate our PL-SLAM method using the classic initialization (based on homography or essential matrix computation), and with the proposed map initialization based only on lines (see again Table I). As expected, the accuracy with the line map initialization drops due to the small rotation

assumptions it does. However, in the low textured sequence *f3_nstr_tex_far*, the classic initialization detects an ambiguity which disables it of initializing the map. In contrast, the proposed line initialization is able to estimate an initial map. In the sequences *f3_walk_xyz* and *f3_walk_halfsph* the proposed initialization does not work due lo large inter-frame rotations produced in the initial frames.

D. Computation Time

While adding line primitives to the visual SLAM improves accuracy and robustness, it also increases the computational complexity. Table II summarizes the time required for each subtask within the "Tracking" and "Local Mapping" blocks, for PL-SLAM and ORB-SLAM [18]. Note that in the subtasks with larger penalties are the map features creation and the local BA. In any event the final frame rate of the PL-SLAM is near real time (20 fps) in a standard and not optimized PC.

VII. CONCLUSIONS

In this work we have proposed PL-SLAM, an approach to visual SLAM that allows to simultaneously process points and lines and tackle situations where point-only based methods are prone to fail, like poorly textured scenes or motion blurred images where feature points are vanished out. We built upon the architecture of the state-of-the-art ORB-SLAM and modify its original pipeline to operate with line features without significantly compromising its efficiency. We have also presented a novel line-based map initialization approach, which estimates camera pose and 3D map from 5 line correspondences in three consecutive images. This approach holds on the assumption of constant and small inter-frame rotation in these three images. In the results section we show that this indeed is a good approximation for many situations. Additionally, we evaluated the full pipeline on the TUM RGB-D benchmark and showed consistent improvement w.r.t. current competing methods.

In future work, we plan to further exploit line features and incorporate other geometric primitives like planes, which can be built from lines in a similar manner as we have built lines from point features.

ACKNOWLEDGMENTS

This work has been partially supported by the EU project AEROARMS H2020-ICT-2014-1-644271, by the MINECO projects RobInstruct TIN2014-58178-R and Rob-Int-Coop DPI2013-42458-P, by the ERA-Net Chistera project I-DRESS PCIN-2015-147 and by the Russian MES grant RFMEFI61516X0003.

REFERENCES

[1] accurate non-iterative o(n) solution to the pnp problem.

[2] N. Ayache and O. D. Faugeras. Building, registrating, and fusing noisy visual maps. *IJRR*, 7(6):45–65, 1988.

[3] S. Bao, M. Bagra, Y. Chao, and S. Savarese. Semantic structure from motion with points, regions, and objects. In *CVPR*, pages 2703–2710, 2012.

[4] A. Concha, W. Hussain, L. Montano, and J. Civera. Incorporating scene priors to dense monocular mapping. *AURO*, 39(3):279–292, 2015.

[5] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *TPAMI*, 29(6):1052–1067, 2007.

[6] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-D mapping with an RGB-D camera. *TRO*, 30(1):177–187, 2014.

[7] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, pages 834–849. Springer, 2014.

[8] O. D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *IJPRAI*, 2(03):485–508, 1988.

[9] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO 2.0: Semi-direct visual odometry for monocular and multicamera systems. *TRO*, 2016.

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012.

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

[12] W. Y. Jeong and K. M. Lee. Visual SLAM with line and corner features. In *IROS*, pages 2570–2575. IEEE, 2006.

[13] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, pages 225–234. IEEE, 2007.

[14] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *ECCV*, pages 802–815, 2008.

[15] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *TPAMI*, 34(7):1381–1393, 2012.

[16] H. Lim, J. Lim, and H. J. Kim. Real-time 6-DOF monocular vision SLAM in a large-scale environment. In *ICRA*, 2014.

[17] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *IMAVIS*, 27(8):1178–1193, 2009.

[18] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular slam system. *TRO*, 31(5):1147–1163, 2015.

[19] R. Newcome and A. J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, pages 1498–1505, 2010.

[20] P. Newman, J. Leonard, J. D. Tardós, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *ICRA*, volume 2, pages 1802–1809. IEEE, 2002.

[21] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *ICRA*, pages 2609–2616, 2014.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRAW*, volume 3, page 5. Kobe, Japan, 2009.

[23] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *ECCV*, pages 430–443, 2006.

[24] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *RSS*, 2009.

[25] P. Smith and I. D. Reid A. J. Davison. Real-time monocular SLAM with straight lines. In *BMVC*, volume 6, pages 17–26, 2006.

[26] J. Sola, T. Vidal-Calleja, J. Civera, and J.M.M. Montiel. Impact of landmark parametrization on monocular EKF-SLAM with points and lines. *IJCV*, 97(3):339–368, 2012.

[27] H. Strasdat, J.M.M. Montiel, and A. Davison. Visual SLAM: Why Filter? *IMAVIS*, 30(2):65–77, 2012.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, Oct. 2012.

[29] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *ISMAR*, pages 209–218. IEEE, 2013.

[30] A. Vakhitov, J. Funke, and F. Moreno-Noguer. Accurate and linear time pose estimation from points and lines. In *ECCV*, 2016.

[31] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD: a line segment detector. *IPOL*, 2:35–55, 2012.

[32] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *CVPR*, pages 1450–1457, 2012.

[33] L. Zhang and R. Koch. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *JVCIR*, 24(7):794–805, 2013.