2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

PL-SVO: Semi-Direct Monocular Visual Odometry by Combining Points and Line Segments

Ruben Gomez-Ojeda, Jesus Briales, and Javier Gonzalez-Jimenez

***Abstract*— Most approaches to visual odometry estimates the camera motion based on point features, consequently, their performance deteriorates in low-textured scenes where it is difficult to find a reliable set of them. This paper extends a popular semi-direct approach to monocular visual odometry known as SVO [1] to work with line segments, hence obtaining a more robust system capable of dealing with both textured and structured environments. The proposed odometry system allows for the fast tracking of line segments since it eliminates the necessity of continuously extracting and matching features between subsequent frames. The method, of course, has a higher computational burden than the original SVO, but it still runs with frequencies of 60Hz on a personal computer while performing robustly in a wider variety of scenarios.**

I. Introduction

Visual odometry (VO) is gaining importance in robotic applications, such as unmanned aerial vehicles (UAVs) or autonomous cars, as an essential part of the navigation systems. Solutions for the VO problem has been addressed employing different sensors, such as monocular or stereo cameras [2] [3] [4], RGB-D cameras [5] [6], or a combination of any of them with an Inertial Measurement Unit (IMU) [7]. The traditional approach consist of the detection and matching of point features between frames, and then, the estimation of the camera motion through least-squares minimization of the reprojection errors between the observed and projected points [8]. In this context, the performance of such approaches deteriorates in low textured scenarios as depicted in Figure 1, where it is difficult to find a large or well-distributed set of image features. In contrast, line segments are usually abundant in human-made scenarios, which are characterized by regular structures rich in edges and linear shapes. Dealing with line segments in images it is not as straightforward as points, since they are difficult to represent [9] and also require high computational burden for the detection and matching tasks thus only a few solutions have been proposed [10] [11], barely reaching real-time specifications. Moreover, edge-based algorithms have been also used for both solving the problem of tracking [12] [13] [14], and estimating the camera motion [15]. However, these methods require a rather costly direct alignment which makes them less suitable to real time, and also limits their application to narrow baseline estimations. To the best of our knowledge, this paper proposes the first real-time approach to Monocular Visual Odometry (MVO) that integrates both point and line segment features, and hence it is capable of working robustly in both structured and textured scenarios. The source code of the developed C++ PL-SVO library and illustrative videos of this proposal can be found here: http://mapir.isa.uma.es

Fig. 1. In low-textured environments, point-based algorithms usually fail due to difficulties in founding a large number of features, in contrast, line segments are usually abundant.

A. Related Work

Visual odometry algorithms can be divided into two main groups. The first one, known as *feature-based*, extract a set of image features (traditionally points) and track them along the successive frames. Then, they estimate the pose by minimizing the projection errors between the correspondent observed features and those projected from different frames. Literature offers us several point-based approaches to the odometry problem, such as PTAM [16], where authors report a fast SLAM system capable of performing real-time parallel tracking and mapping over thousands of landmarks. In contrast, the problem of motion estimation with line features has been less explored due to their inherent difficulties, specially to monocular odometry. In [17] authors extend the Iterative Closest Point (ICP) approach [18] to the case of stereo odometry with line segments, where they substitute the computation of costly descriptors in a one-to-multiple line matching approach. In our previous work [19], we present a stereo visual odometry system based on both point and line segment features. The influence of each feature is weighted with the inverse of their covariance matrix, which is obtained by uncertainty propagation

Mapir Group. Universidad de Málaga. E.T.S. Ingeniería de Informática. Campus de Teatinos, 29071, Málaga, Spain. E-mail: rubengooj@gmail.es

This work has been supported by the Spanish grant program FPU14/06098 and the project "PROMOVE: Advances in mobile robotics for promoting independent life of elders", both funded by the Spanish Government and the "European Regional Development Fund ERDF" under contract DPI2014-55826-R.

techniques over the reprojection functions. However, this work still relies on traditional feature detection and matching, and thus it has a high computational cost.

The other group, known as *direct* approaches, estimates the camera motion by minimizing the photometric errors between successive frames at several image locations. In [20] authors propose a direct approach, known as DTAM, where they estimate the camera pose by direct alignment of the complete intensity image between each keyframe, employing a dense depth-map estimation. However, this method requires GPU parallelization since it process the whole image. For dealing with the high computational requirements of direct methods, a novel monocular technique is proposed in [21], where authors estimate the camera motion in a semi-dense approach, thus reaching real-time performance on a CPU. They continuously estimate and track a semi-dense inverse depth-map for regions with a sufficient image gradient, thus only exploiting those areas which introduce valid information to the system. Then, they estimate the camera motion by minimizing the photometric error over the regions of interest, hence combining the good properties of direct algorithms with a sparse approach that allows for fast processing.

B. Contributions

Point features are less abundant in low-textured and structured scenarios, and hence, the robustness and accuracy of point-based visual odometry algorithm dramatically decreases. On the other hand, detecting and matching line segments demands high computational resources, which is the main reason of the lack of real-time approaches to visual odometry using these features. In this work, we extend the semi-direct approach in [1] to the case of line segments, which performs fast feature tracking as an implicit result of a sparse-direct motion estimation. Therefore, we take advantage of this sparse structure to eliminate costly segment detection (we only detect them when a new keyframe is introduced to the framework), and descriptor computation, while maintaining the good properties of line segments. As a result, we contribute with a fast monocular odometry system capable of working robustly in low textured scenarios thanks to the combination of the information of both points and segment features. In the following we describe the proposed system, and validate the claimed features with experiments in different environments.

II. System Overview

The proposed system can be understood as an extension of the semi-direct framework in [1], that not only consider points but also segment features in the scene and introduce both in the pipeline. This is a non-trivial extension, since line segments present more complexity than point features from a geometrical point of view. In practice, this makes that certain image operations which

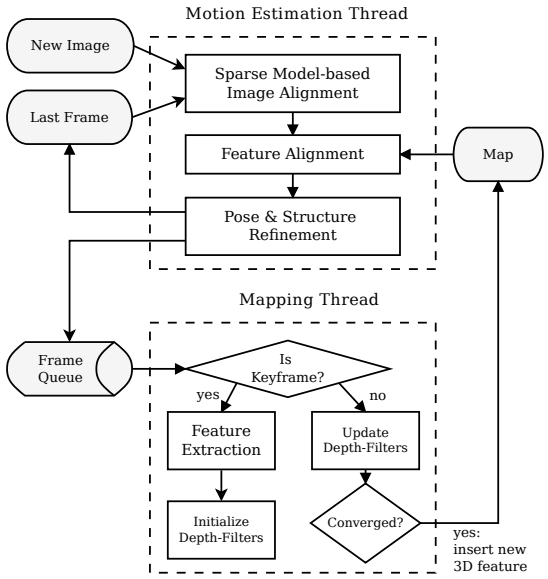


Fig. 2. SVO framework, extracted from [1]. Our work extends the concept of feature so that both points and segments in the scene are considered for every step in the pipeline.

are almost trivial for points become more computationally cumbersome for the case of segments. Hence, we need to perform several approximations and take some well-founded heuristic in order to save computational resources. These will be seen in higher detail in the Sections III and IV.

For the sake of completeness, we briefly review every stage of the semi-direct framework [1], depicted in Figure 2 while showing how the partnering of this semi-direct approach and the use segment features becomes mutually beneficial. The semi-direct approach is divided into two parallel threads, one for estimating the camera motion and another one for mapping the environment.

In the motion thread, an initial motion estimate is performed between consecutive frames by using a sparse direct alignment approach (see Figure 3), which minimizes the photometric error between patches using the 3D warping provided by the known 3D features. This allows for the fast tracking of features between frames as a result of the semi-direct motion estimation, which eliminates the need of performing frame-to-frame detection and matching. This, which is fairly advantageous for point features, becomes extremely beneficial for segments since they are considerably more computationally expensive. Instead, features are only detected when a new keyframe is inserted, so that the overall cost of the LSD segments detection [22] becomes affordable. Furthermore, by reducing the dimension of the optimization problem to the estimation of the pose only epipolar geometry is automatically fulfilled and we do not have to take care of outlier matches.

Then, the second step (see Figure 4) of the motion thread is to refine the feature projections given by the transformation estimate from direct alignment, thus violating the epipolar constraints to reduce the drift of the

camera. The feature refinement is performed by taking as reference patch the one with the closest viewpoint. This approach, again, is very beneficial for segments since it limits large observations baselines during the tracking of the segments. In consequence, it alleviates well-known issues of line segments such as endpoints repeatability, occlusions or deformation of the segments due to change of view [11]. Finally, both the camera motion and the map structure are refined by minimizing the reprojection errors.

At this point, the matching with far features is fully solved thanks to the intermediate continuous tracking and we can apply specific feature-based refinement approaches that behave quite well for segment features, as depicted in Figure 5.

To sum up, we see that the introduction of segments in a semi-direct framework [1] can be done much more seamlessly that for other more traditional approaches, since the preliminary direct steps alleviate most of the downsides that have historically prevented the use of segments in Visual Odometry. Otherwise stated, the motion as well as the mapping are enriched by the use of segments without incurring in a significant overhead of the overall system.

Concurrently, the map thread estimates the depth of 2D features with a probabilistic Bayesian filter, which is initialized when a keyframe is inserted to the pipeline. The depth filters are initialized with a high uncertainty, but they converge to the actual values in a few iterations and are then inserted to the map, becoming useful for motion estimation. In the following, we describe in detail each stage of the algorithm, and then validate it with experiments in real environments.

III. Semi-Direct Monocular Visual Odometry

Let C_{k-1} and C_k be the coordinate systems of a calibrated camera at two consecutive poses, which are related by the relative pose transformation $\mathbf{T}_{k-1,k}(\boldsymbol{\xi}) \in SE(3)$, where $\boldsymbol{\xi} \in \mathfrak{se}(3)$ is the 6-vector of coordinates in the Lie algebra $\mathfrak{se}(3)$. The problem we face is that of estimating the camera pose along a sequence of frames, for which we denote $\mathbf{T}_{k,w}$ as the camera pose with respect to the world's reference system in the k-th timestep. For that, let us denote as I_k the intensity image in the k-th frame, and $\boldsymbol{\Omega}$ as the image domain. We will denote the point features as $\mathbf{x}$, and its correspondent depth as $d_{\mathbf{x}}$. In the case of line segments, we will employ both the endpoints, denoted by $\mathbf{p}$ and $\mathbf{q}$ respectively, and the line equation as $\mathbf{l}$. The 3D point back-projected from the image at timestep k is denoted as $\mathbf{X}_k$, and can be obtained through the inverse projection function $\boldsymbol{\pi}^{-1}$, i.e. $\mathbf{X}_k = \boldsymbol{\pi}^{-1}(\mathbf{x}, d_{\mathbf{x}})$. The projection of a 3D point in the image domain is obtained through the camera projection model $\boldsymbol{\pi}$, so that $\mathbf{x} = \boldsymbol{\pi}(\mathbf{X}_k)$. In the following, we will extend the steps of SVO algorithm to the case of line segments.

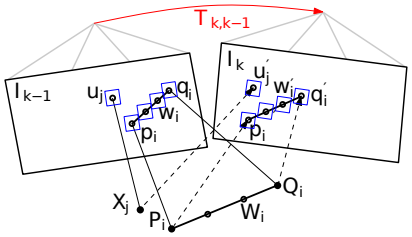


Fig. 3. The relative pose between the current and the previous frame parameterizes the position of the reprojected points in the new image. We perform (sparse) image alignment to find the pose that minimizes the photometric difference between image patches corresponding to the same 3D point (blue squares). For the segments points are homogeneously sampled between the 3D endpoints. Note, in all figures, the parameters to optimize are drawn in red and the optimization cost is highlighted in blue. This figure has been adapted from [1].

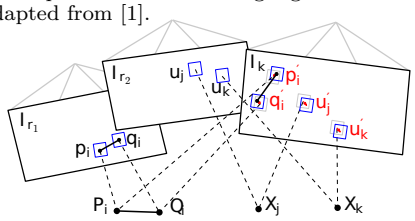


Fig. 4. The 2D position of each point is optimized individually to minimize the photometric error in its patch. For the segments the end points are similarly optimized. This alleviates errors propagated from map and camera pose estimation. This figure was also adapted from [1].

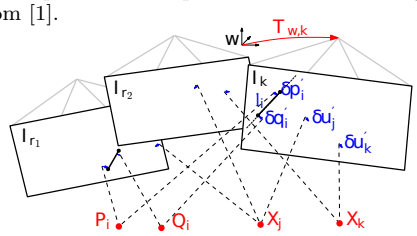


Fig. 5. In the last motion estimation step, the camera pose and the structure (3D points and segments) are optimized to minimize the reprojection error that has been established during the previous feature-alignment step. Similarly to the previous ones, this figure has been adapted from [1].

A. Sparse Model-based Image Alignment

The camera motion between two consecutive frames, $\mathbf{T}_{k-1,k}(\boldsymbol{\xi})$, is first estimated through direct image alignment of the sparse features tracked along the frames. Unlike the point-based approach, we cannot directly align the whole region occupied by line segment between two frames, since it would be computationally expensive. For that, we only minimize the image residuals between some patches equally distributed all along the line segment, as depicted in Figure 3. Let us define $\mathcal{L}$ as the image region for which the depth of the endpoints is known at previous time step $k-1$, and for which the endpoints $\mathbf{p}$ and $\mathbf{q}$ are visible in the image domain at the current timestep $\boldsymbol{\Omega}_k$:

$$\begin{aligned}\mathcal{L} := \big\{ \mathbf{p}, \mathbf{q}, \mathbf{w}_n \mid \mathbf{p}, \mathbf{q} \in \mathcal{L}_{k-1} \\ \wedge\ \boldsymbol{\pi}\big(\mathbf{T}(\boldsymbol{\xi}) \cdot \boldsymbol{\pi}^{-1}(\mathbf{p}, d_{\mathbf{p}})\big) \in \boldsymbol{\Omega}_k \\ \wedge\ \boldsymbol{\pi}\big(\mathbf{T}(\boldsymbol{\xi}) \cdot \boldsymbol{\pi}^{-1}(\mathbf{q}, d_{\mathbf{q}})\big) \in \boldsymbol{\Omega}_k \big\} \end{aligned} \quad (1)$$

where $\mathbf{w}_n$, with $m = 2, ..., N_l - 1$ referring to the intermediate points defined homogeneously along the line

segments.

Then, the intensity residual for a line segment δI_l is defined as the photometric difference between pixels of the same 3D line segment point, which is:

$$\delta I_l(\boldsymbol{\xi}, \mathbf{l}) = \frac{1}{N_l} \sum_{n=0}^{N_l} \left| I_k\Big(\boldsymbol{\pi}\big(\mathbf{T}(\boldsymbol{\xi}) \cdot \mathbf{w}_n\big)\Big) - I_{k-1}(\mathbf{w}_n) \right| \quad (2)$$

where in the case of $n = 0$ and $n = N_l$, the point $\mathbf{w}_n$ refers to the endpoints $\mathbf{p}$ and $\mathbf{q}$ respectively. Then, we estimate the optimal pose increment $\boldsymbol{\xi}^*_{k-1,k}$ that minimizes the photometric error of all patches, for both point and line segment features:

$$\boldsymbol{\xi}^*_{k-1,k} = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \Big\{ \sum_{i \in \mathcal{P}} \|\delta I_p(\boldsymbol{\xi}, \mathbf{x}_i)\|^2 + \sum_{j \in \mathcal{L}} \|\delta I_l(\boldsymbol{\xi}, \mathbf{l}_j)\|^2 \Big\}. \quad (3)$$

Similarly to [1], we employ inverse compositional formulation proposed in [23], for speeding up the minimization process. In this case, we seek for the linearized Jacobian of the line segment residuals, which can be expressed as the summatory of the individual point Jacobians for each intermediate point $\mathbf{w}_n$ sampled:

$$\left.\frac{\partial \delta I_l(\boldsymbol{\xi}, \mathbf{l}_j)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} = \frac{1}{N_l} \sum_{m=0}^{N_l} \left.\frac{\partial \delta I_p(\boldsymbol{\xi}, \mathbf{w}_n)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} \quad (4)$$

whose expression can be obtained of [1]. Then, we estimate the optimal pose by robust Gauss-Newton minimization of the above-mentioned cost function in (3). Notice that this formulation allows for the fast tracking of line segments as depicted in Figure 1, which is an open problem due to the high computational burden employed with traditional feature-based approaches [19].

B. Individual Feature Alignment

Similarly to [1], we individually refine the 2D positions of each feature by minimizing the photometric error between the patch in the current image, and the projection of all the 3D observations of this feature, which can be solved by employing Lucas-Kanade algorithm [23]. In the case of line segments, we only need to refine the position of the 2D endpoints (see Figure 4), which defines the line equation employed in the estimation of the projection errors:

$$\mathbf{w}'_j = \underset{\mathbf{w}'_j}{\operatorname{argmin}} \left\| I_k(\mathbf{w}'_j) - I_r(\mathbf{A}_j \cdot \mathbf{w}_j) \right\|^2, \ \forall j \quad (5)$$

where $\mathbf{w}'_j$ is the 2D estimation of the position of the feature in the current frame ($\mathbf{w}'_j$ stands equally for both endpoints), and $\mathbf{w}_j$ is the position of the feature in the reference frame r. This is a bold assumption in the case of line segments, since their endpoints are considerably less descriptive than keypoints. For dealing with this, we also perform a robust optimization of (5), and then we relax this assumption by refining the 3D position of the endpoints. Notice that it is necessary to employ an affine warping $\mathbf{A}_j$ in this step, since the closest key frame for which we project the feature is usually farther, and the size of the patch is bigger than in the previous step.

C. Pose and Structure Refinement

After optimizing individually the position of each feature in the image by skipping the epipolar constraints, the camera pose obtained in (3) must be further refined by minimizing the reprojection errors between the 3D features and the corresponding 2D feature positions in the image (see Figure 5). For that, we consider reprojection errors between the 3D features and the camera pose $T_{k,w}$, both in world's coordinate frame, since it considerably reduces the drift of the estimated trajectory. The cost function when employing both type of features is:

$$\boldsymbol{\xi}^*_{k,w} = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \Big\{ \sum_{i \in \mathcal{P}} \|r_p(\mathbf{T}_{k,w}, \mathbf{X}_{i,k})\|^2 + \sum_{j \in \mathcal{L}} \left\|r_l(\mathbf{T}_{k,w}, \mathbf{P}_{j,k}, \mathbf{Q}_{j,k}, \mathbf{l}_j)\right\|^2 \Big\} \quad (6)$$

where r_p stands for the projection errors in the case of point features, and r_l is the projection error of line segments:

$$r_l(T_{k,w}, \mathbf{P}_{j,k}, \mathbf{Q}_{j,k}, \mathbf{l}_j) = \begin{bmatrix} \mathbf{l}_j \cdot \boldsymbol{\pi}\big(\mathbf{T}_{k,w} \cdot \mathbf{P}_{j,k}\big) \\ \mathbf{l}_j \cdot \boldsymbol{\pi}\big(\mathbf{T}_{k,w} \cdot \mathbf{Q}_{j,k}\big) \end{bmatrix}. \quad (7)$$

This is solved iteratively with Gauss-Newton, for which we need to include (6) in a robustified framework, for which we employ the Cauchy loss function:

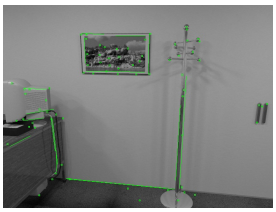
$$\rho(s) = log(1 + s). \quad (8)$$

The optimization consist of three steps: i) we first estimate the camera motion with all the samples ii) we filter out the outliers, which are considered as those features whose residual error lies above two times the robust standard deviation of those errors iii) we fastly refine the camera pose by optimizing with the inliers subset. Finally, we refine the position of the 3D point and line segment features through minimization of the reprojection errors.

IV. Mapping

The map thread recursively estimates the 3D position of the image features for which their depth is still unknown. For that, authors of [1] implement a depth filter based in a Bayesian framework, for which they model the depth of the feature with a Gaussian + Uniform mixture model distribution [24]. In the case of line segments we need to estimate the 3D position of the endpoints, since they are employed for both describe the feature and estimating the reprojection errors. However, the endpoints of line segments obtained through detectors such as LSD [22] are not repetitive, which is a limitation to employ them in monocular visual odometry. On the other hand, one of the advantages of the fast tracking employed here is that we explicitly seek for the exact same line segment in the successive frames, so that we continuously track the position of the endpoints. This allows for the introduction of the endpoints in a similar

(a) Living room



(b) Office

Fig. 6. Sparse features tracked by PL-SVO in two frames extracted from the ICL-NUIM dataset [25], where we can observe the importance of introducing line segments in such low-textured scenarios.

Bayesian framework, where the distribution of both endpoints is estimated when inserting new observations. As a result, we obtain meaningful maps which can be used to extract useful information about geometry of the scene.

V. Experimental Validation

In this section, we illustrate the benefits of including line segments in motion estimation, specially when working in low-textured environments. For that, we estimate the trajectory of a monocular camera in several sequences, from both synthetic and real datasets. All experiments have been conducted on an Intel Core i5-6600 CPU @ 3.30GHz without GPU parallelization.

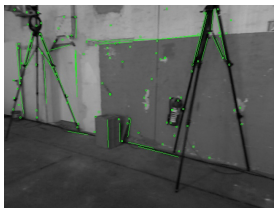
A. Evaluation in ICL-NUIM Dataset [25]

First, we test our algorithm in the Imperial College London and National University of Ireland Maynooth (ICL-NUIM) dataset [25]. This dataset consist of two different synthetic environments, one in an office and the other one in a living room, for which several sequences can be generated and rendered (see Figure 6). Table III compares the performance of the proposed algorithm against SVO [1] for the sequences *lrkt-2* and *ofkt-3*. For the sake of fairness, we have employed our current implementation of PL-SVO without introducing line segments to the framework as baseline of comparison. Results show a superior performance of PL-SVO in the first sequence *lrkt-2*, which is capable of estimating the camera motion along the whole trajectory (the rest of the sequence is employed for initialization), while the point-based approach only tracks the 34% of the sequence. In the second sequence, *ofkt-3*, SVO shows a slight superiority in terms of accuracy. However, it is worth noticing that it is only capable of tracking a 57%, and hence, it is not affected by higher errors introduced in the difficult parts of the sequence.

B. Evaluation in TUM Dataset [26]

We also evaluate the performance of both SVO and PL-SVO approaches in the TUM Dataset [26], which consist of several sequences recorded with an RGB-D camera in different environments, as depicted in Figure 7. Table II contains the results for the considered sequences from the TUM dataset. In general, we observe the superior performance of PL-SVO in most sequences, hence confirming its robust behavior in multiple environments.

(a) Textured scene



(b) Low-textured scene

Fig. 7. Sparse features tracked by PL-SVO in two different frames of the TUM dataset [25].

However, the accuracy of motion estimation considerably decreases in this dataset, where monocular techniques are severely affected by motion blur and other negative effects resulting from the rolling shutter in RGB-D sensors.

C. Processing Time

Finally, we analyze the impact of introducing line segments to the framework in the processing time. Table I shows the average times employed in the different stages of the algorithms. As one may first think, the computational cost necessary for performing both sparse image and feature alignment increases considerably when including line segments, where the runtime of each stage is augmented in 4 ms. However, our algorithm still performs in real-time with frequencies of almost 60 Hz, depending on the type of scene.

TABLE I
Mean average times in each stage of the algorithm for both SVO and PL-SVO algorithms.

	SVO [1]	PL-SVO
Pyramid creation	0.26 ms	0.26 ms
Sparse Image Alignment	2.60 ms	6.58 ms
Feature Alignment	4.13 ms	8.61 ms
Pose and Structure Refinement	0.35 ms	0.76 ms
Total Motion Estimation:	**8.60 ms**	**17.83 ms**

VI. Conclusions

In this paper we have proposed a novel approach to monocular odometry by extending the SVO algorithm proposed by Forster et al. in [1] to the case of line segments. Hence, we obtain a more robust system capable of dealing with untextured environments, where performance of point-based approaches usually deteriorates due to the difficulties in finding a well-distributed set of points. The semi-dense approach allows for the fast tracking of line segments, thus eliminating the necessity of detecting and matching whenever a new frame is introduced, which is one of the main limitations of employing this type of features. We validate the claimed features in a series of experiments in both synthetic and real datasets, confirming the robust behavior of this proposal.

References

[1] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," *IEEE International Conference on Robotics and Automation*, 2014.

[2] G. Klein and D. Murray, "Improving the agility of keyframe-based slam," in *Computer Vision–ECCV 2008*, pp. 802–815, Springer, 2008.

TABLE II

COMPARISON AGAINST SVO [1] IN THE ICL-NUIM DATASET BY MEASURING RELATIVE POSE ERRORS (RPE) PER SECOND.

		% Sequence	RMSE (m/s)	Median (m/s)	RMSE (deg/s)	Median (deg/s)
lrkt-2	SVO	34.32	0.0085	0.0078	**0.0615**	**0.0491**
	PL-SVO	**91.93**	**0.0076**	**0.0069**	0.1023	0.0502
ofkt-3	SVO	57.82	**0.0053**	**0.0041**	0.1011	0.0547
	PL-SVO	**96.85**	0.0059	0.0053	**0.0997**	**0.0532**

TABLE III

COMPARISON AGAINST SVO [1] IN THE TUM DATASET BY MEASURING THE RPE PER SECOND.

		% Sequence	RMSE (m/s)	Median (m/s)	RMSE (deg/s)	Median (deg/s)
fr1-floor	SVO	54.47	0.4112	**0.0528**	21.0040	2.1970
	PL-SVO	**77.11**	**0.0806**	0.0742	**2.2658**	**1.1294**
fr1-xyz	SVO	95.10	0.1780	0.1251	11.8003	8.8365
	PL-SVO	95.10	**0.1089**	**0.0873**	**7.6256**	**5.9863**
fr2-desk	SVO	96.23	0.0908	0.0828	0.9912	0.7675
	PL-SVO	96.23	**0.0693**	**0.0644**	**0.9040**	**0.7275**
fr2-rpy	SVO	98.39	**0.0155**	**0.0100**	**0.6424**	**0.5037**
	PL-SVO	98.39	0.0157	0.0107	0.6501	0.5200
fr2-xyz	SVO	98.56	0.0213	0.0183	0.6462	0.5449
	PL-SVO	98.56	**0.0209**	**0.0178**	**0.6337**	**0.4845**
fr3-longoffice	SVO	95.35	0.1794	0.1793	**1.1132**	0.6138
	PL-SVO	95.35	**0.1660**	**0.1637**	1.3118	**0.5161**

[3] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," *Image and Vision Computing*, vol. 27, pp. 588–596, apr 2009.

[4] M. Persson, T. Piccini, M. Felsberg, and R. Mester, "Robust stereo visual odometry from monocular techniques," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 686–691, IEEE, 2015.

[5] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 3748–3754, IEEE, 2013.

[6] M. Jaimez and J. Gonzalez-jimenez, "Fast Visual Odometry for 3D Range Sensors,"

[7] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI*, no. EPFL-CONF-214687, 2015.

[8] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *Robotics & Automation Magazine, IEEE*, vol. 18, no. 4, pp. 80–92, 2011.

[9] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[10] M. Chandraker, J. Lim, and D. Kriegman, "Moving in stereo: Efficient structure and motion using lines," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1741–1748, IEEE, 2009.

[11] T. Koletschka, L. Puig, and K. Daniilidis, "MEVO: Multi-environment stereo visual odometry," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4981—-4988, sep 2014.

[12] C. Harris and C. Stennett, "Rapid-a video rate object tracker.," in *BMVC*, pp. 1–6, 1990.

[13] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3d camera tracking," in *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pp. 48–56, IEEE, 2004.

[14] G. Reitmayr and T. Drummond, "Going out: robust model-based tracking for outdoor augmented reality," in *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 109–118, IEEE Computer Society, 2006.

[15] M. Kuse and S. Shen, "Robust camera motion estimation using direct edge alignment and sub-gradient method," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 573–579, May 2016.

[16] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[17] J. Witt and U. Weltin, "Robust Stereo Visual Odometry Using Iterative Closest Multiple Lines," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4164—-4171, IEEE/RSJ, 2013.

[18] D. G. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 441–450, 1991.

[19] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016.

[20] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327, IEEE, 2011.

[21] J. Engel, J. Sturm, and D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 1449–1456, IEEE, 2013.

[22] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[23] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On : A Unifying Framework : Part 1 2 Background : Lucas-Kanade," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[24] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, pp. 434–441, 2011.

[25] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, (Hong Kong, China), May 2014.

[26] 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)*, 2012.