Sparse Point-Plane SLAM

Mehdi Hosseinzadeh, Yasir Latif, Ian Reid
University of Adelaide, Australia
firstname.lastname@adelaide.edu.au

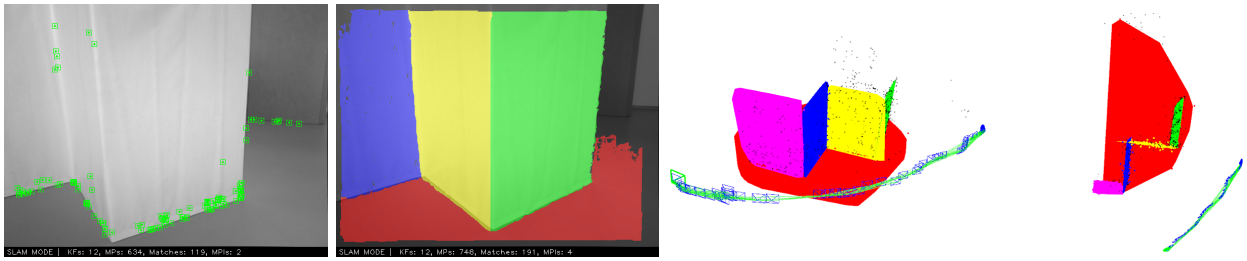


Figure 1: Sparse Point-Plane SLAM results on the challenging low-texture `fr3/str_notex_far` sequence: **(1)** Original ORB-SLAM frame with tracked points, **(2)** Detected Planes. Map obtained from points and planes, **(3)** Side view and **(4)** Top view. Considering planes in spare mapping leads to more meaningful map representations.

Abstract

SLAM is a fundamental problem of mobile robotics and its most efficient version models the environment as set of sparse points, against which the camera is tracked. We explore the possibility of using geometric information in the form of planes to further improve the accuracy of tracking and mapping. Plane measurements are considered and planar constraints are introduced between points and planes on which they lie. We integrate planar measurements and constraints in a state-of-the-art feature-based SLAM system and show that this provides better trajectory estimates than points alone, and more meaningful maps, especially in environment with low texture and dominantly planar surfaces.

1 Introduction

Simultaneous Localization And Mapping (SLAM) is considered one of the fundamental challenges in mobile robotics [Cadena *et al.*, 2016] and addresses the problem of building the representation (map) of a previously unseen environment while simultaneously localizing a mobile robot with respect to it. While the representation of the robot depends on the degrees of freedom of motion, the representation of the map depends on many factors such as the intended high level task, the available sensors, and computational resources. Many possible representations have been presented in literature.

In its simplest form, the map can be represented as a collection of sparse landmarks (points), each of which corresponds to a salient region in an image. This representation has the advantage of being computationally efficient due to its sparse nature. Point based methods have been successfully used to map city-scale environments. However, this sparsity comes at a price: such maps lack semantic information and are not useful for high level task such as grasping and manipulation.

A more computationally demanding but meaningful approach is to consider all pixels in the image that have significant gradient and to use them as candidates for points in the map. This leads to so called "semi-dense" mapping, where the structure of the environment is more interpretable and can be used for some high level tasks. Semi-dense system have been shown to work at a building-scale environment successfully.

At the extreme end of the density scale lies a fully dense map representation where the environment is represented as a collection of surfaces. This contains the most information for tasks such as manipulation, while being the most expensive to compute. Dense mapping is still limited to room-scale environments.

A map based on sparse points carries no semantic in-

formation. Geometric entities such as lines and planes, on the other hand, carry semantic information such as edges or surfaces present in the scene. These primitive as useful for map representation as they provide informative constraints on the scene geometry as well as the robot's pose, while still being sparse.

Man-made environments contain many planar surfaces and it is easy to infer relationships between them using a Manhattan world assumption: the planes are either parallel or perpendicular to each other. Detecting these planes and using them in addition to a point-based representation provides additional constraints for the points that lie on one of the planes and in-turn inform the camera pose. These planar constraints are also useful in areas where features can not be found because of uniform texture. In this work, we explore the effect of using planes extracted from RGB-D sensor data to further constrain a sparse environment representation. Planes are extracted from RGB-D data and integrated into a state-of-the-art sparse mapping system. The extracted planes are used for both localization and mapping.

We show that this greatly reduces the error in the estimated camera trajectory without incurring great computational cost. A point and plane based representation is richer and more informative than a sparse point representation and can be used for reasoning about semantics such as object affordances.

The rest of the paper is organized as followed. In the next section, we present a short overview of methods that have already been developed to address the problem of SLAM based on a combination of points and other geometric primitives. Section 3 presents an overview of our system and experiments are presented in Section 4. Finally, the conclusion and future research directions are presented.

2 Related Work

SLAM is well studied problem in mobile robotics and many different solution have been proposed for solving it. The most recent of these is the graph-based approach that formulates SLAM as a nonlinear least squares problem [Grisetti *et al.*, 2010]. SLAM with cameras has also seen advancement in theory and good implementations that have lead to many real-time systems from sparse ([Mur-Artal *et al.*, 2015],[Engel *et al.*, 2017]) to semi-dense ([Engel *et al.*, 2014], [Forster *et al.*, 2014]) to fully dense ([Newcombe *et al.*, 2011b], [Newcombe *et al.*, 2011a], [Prisacariu *et al.*, 2017]).

We are interested in sparse map representations incorporate geometric primitives such as lines and/or planes, without focusing on the modality of the sensor used (monocular vs. RGB-D). Recently, there has been a lot of interest in extending the capability of a point-based representation by either applying the same techniques to other geometric primitives or fusing points with lines or planes to get better accuracy. Several methods have explored replacing points with lines [Lemaire and Lacroix, 2007], [Gee and Mayol-Cuevas, 2006]. However, lines present especial difficulty because of the lack of a good mathematical representation that is amenable to the least-squares frame-work. Some works have explored the possibility of using lines and points in the same framework [Pumarola *et al.*, 2017], [Gomez-Ojeda *et al.*, 2017] and have been more successful.

Recently, [Kaess, 2015] proposed a representation for infinite planes that is amenable for use in a least-squares framework. Using this representation, they presented a method that work using just information of planes visible in the environment. Similarly, [Yang *et al.*, 2016] use a monocular input to generate plane hypothesis using a Convolutional Neural Network (CNN) which is then refined over time using both the planes as well points in the images. [Taguchi *et al.*, 2013] proposed a method that fuses points and planes using an RGB-D sensor. In the latter works, they try to fuse the information of plane entities to increase the accuracy of depth inference.

In this work, we present a complete pipe-line that take build on top of a state-of-the-art sparse SLAM system by introducing plane landmarks and constraints between planes and points. In the experiments section, we show that the accumulated trajectory error is greatly reduced by introduction of plane primitive for tracking and mapping.

3 Point-Plane SLAM

In order to take advantage of the recent success of graph-based approaches to SLAM, we build our system on top of the state-of-the-art feature-based SLAM framework, ORB-SLAM [Mur-Artal *et al.*, 2015]. This allows us to benefit from the sparsity of the map and at the same time incorporate more semantically meaningful geometric primitives such as planes in the map.

Modern SLAM algorithm can be divided into two parts: the front-end, that deals with sensors to convert them into constraints, and a back-end, that can optimize over these constraints in a least-squares framework. In order to incorporate planes as landmarks in a SLAM framework, both the front-end and the back-end of the SLAM system have to be updated. Fig 2 represents the whole pipeline of our point-plane SLAM framework. In this pipeline, plane measurement and matching part is done at the front-end and optimization based on planes and points is carried out by the back-end.

In the following, we provide detailed explanation of each of these components.

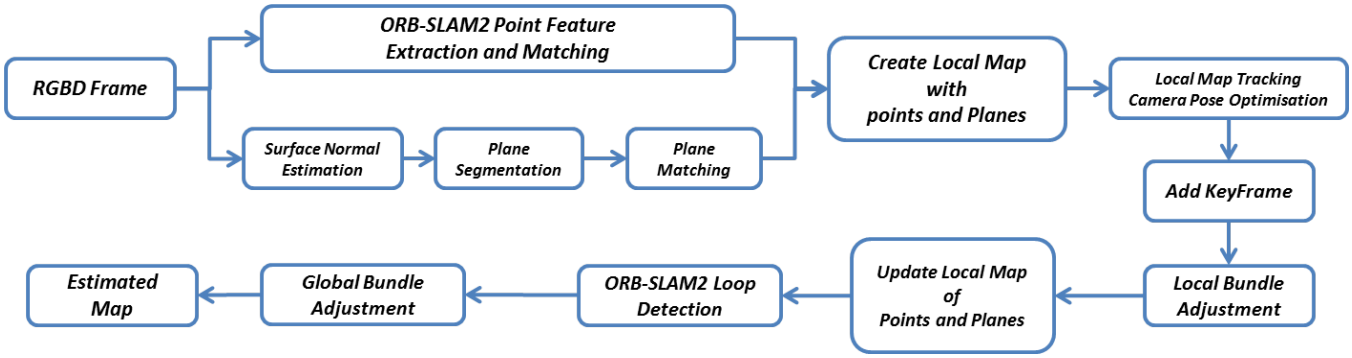


Figure 2: The pipeline of our sparse point-plane SLAM system.

3.1 Front-end of Point-Plane SLAM

A. Point landmarks

The front end for feature selection and matching comes from ORB-SLAM2 and we do not make any changes to it. ORB-SLAM extracts and matches ORB features in a coarse-to-fine pyramid. As illustrated in Fig 2, points are tracked after extracting and matching ORB descriptors in each frame, which is extremely fast to compute than SIFT and SURF features.

B. Plane landmarks

Most of the plane-based SLAM frameworks extract planes from the RGB-D data by fitting planar models using RANSAC which is extremely slow for our purpose of building a near real-time SLAM framework with point and plane landmarks.

Plane Segmentation: We are interested in detecting infinite planes from RGB-D data along with the finite boundary of the plane in each frame. This information will be used to associate points to the detected infinite planes and match planes across images. For this purpose our plane segmentation follows [Trevor *et al.*, 2013] which efficiently segments point clouds from RGB-D data in near real-time.

For point cloud segmentation, we calculate surface normals for each valid depth measurement by smoothing in its neighbourhood based on the proposed method of [Holzer *et al.*, 2012]. After estimating the surface normals, connected component labelling starts with the comparison of:

$$n_1.n_2 < \theta_{thresh} \ , \ |d_1 - d_2| < d_{thresh} \tag{1}$$

where n_i and d_i are surface normal vectors and depth of two points respectively and θ_{thresh} and d_{thresh} are corresponding thresholds on angle and distance. This comparison function yields plane segmentation of the point cloud.

Data Association and Plane Matching: Planes detected in a frame need to be tracked into the subsequent frame and matched with planes detected in the new frames. Due to the inherent robust structure of these landmarks, we use geometric information of the detected planes for data association. Planes are matched based on the difference between their normals and the euclidean distance between them. The best match is selected based on these measures after removing the outliers by thresholding.

3.2 Back-end of Point-Plane SLAM

We formulate our point-plane mapping problem, as a graph-based optimization problem and use the benefits of the least squares estimation problems. In our problem, we want to estimate the F camera poses w.r.t to world reference frame, $\mathbf{T}_1^w, \mathbf{T}_2^w, ..., \mathbf{T}_F^w$, 3D points in the map $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ and infinite planes in the map $\pi_1, \pi_2, ..., \pi_L$ using the point and plane measurements in the frames.

We represent the mentioned estimation problem as a factor graph shown in Fig 3. In this graph, 3D point $\mathbf{x}_i$ is represented by its homogeneous coordinates and camera pose is represented as an isometric 3D Euclidean transformation $\mathbf{T}_c^w$ between a global coordinates frame and local camera coordinates frame. The measurement of a 3D point in a local camera frame is calculated based on geometric re-projection error as in the traditional point-based ORB-SLAM2.

To integrate plane landmarks in this graph, we represent a plane π_i by its homogeneous coordinates $\pi_i = [a\ b\ c\ d]^T$ with normal vector $\vec{\mathbf{n}} = [a\ b\ c]^T$ and distance to origin d. Due to the following relationship between homogeneous coordinates of a point $\mathbf{x}_i$ which belongs to a plane π_i, [Hartley and Zisserman, 2003],

$$\pi_i{}^T \mathbf{x}_i = 0 \tag{2}$$

we can find the transformation between local camera and global coordinates frames of points and planes with the relations $\mathbf{x}_w = (\mathbf{T}_c^w)^{-1}\mathbf{x}_c$ and $\pi_w = (\mathbf{T}_c^w)^T \pi_c$ which sub-

script c denotes local camera frame and w world global frame.

To solve the nonlinear least squares problem associated with the factor graph Fig. 3 by nonlinear gradient-based methods, we need a minimal representation for the planes since the over-parameterised representation makes the information matrix rank-deficient.

To overcome this problem, inspired by [Kaess, 2015] we represent our plane landmarks by their normalised homogeneous vectors as their minimal representation (3 DOF) which forms a 3 dimensional manifold $\mathcal{S}^3$. We consider the northern hemisphere of $\mathcal{S}^3$ as our plane representation manifold (since antipodal points both represent a same plane landmark). This can be represented by the $\mathbb{SO}(3)$ Lie group as our manifold representation of our planes or its isomorphic group of quaternions. So instead of solving a constrained optimisation problem on $\mathcal{S}^3$, we can update planes during optimisation by using the exponential map on the mentioned manifold.

By this formulation of the planes, the factor between a plane and camera pose is

$$\| \, d(\mathbf{T}_c^{w\,-T} \tilde{\pi_w}, \pi_c) \, \|_{\Sigma}^2 \tag{3}$$

squared Mahalanobis norm with covariance matrix $\mathbf{\Sigma}$ of the difference d in the tangent space between plane estimation $\tilde{\pi_w}$ and plane actual measurement π_c . For more details regarding plane updates and their corresponding exponential map refer to [Kaess, 2015].

A factor between point and plane is a constraint which models an inlier points of the plane using an orthogonal Euclidean distance with variance σ of the point $\mathbf{x}$ from the infinite plane by

$$\| \, \vec{\mathbf{n}}^T(\mathbf{x} - \mathbf{x}_o) \, \|_{\sigma}^2 \tag{4}$$

where $\vec{\mathbf{n}}$ is a unit normal vector of the infinite plane and $\mathbf{x}_o$ is one point on the infinite plane. Note that the association between inlier points and planes have been established in the front-end during plane detection and segmentation.

Planes during optimisation Since the number of infinite planes observable in each frame is much less than the number of interest points, integrating planes in the optimisation does not introduce additional computational cost on top of optimisation for point landmarks. Introducing planes has significant effect on *a) camera localization wrt a keyframe* and *b) local bundle adjustment over the landmarks in the map including points and planes after adding a new keyframe.*

In the camera localization stage, the observed landmarks are assumed to be fixed and the pose of the camera is optimized according to the factor graph Fig 3. Using planes in this stage improves the accuracy of the camera tracking significantly which is shown in the section 4. When a new keyframe is added by ORB-SLAM2, local bundle adjustment is carried out by optimising a pose graph consisting of the new keyframe and adjacent keyframes which have observed the common points and planes landmarks. This stage also has a significant part in global consistency of the reconstructed map and its effect can be seen in Section 4

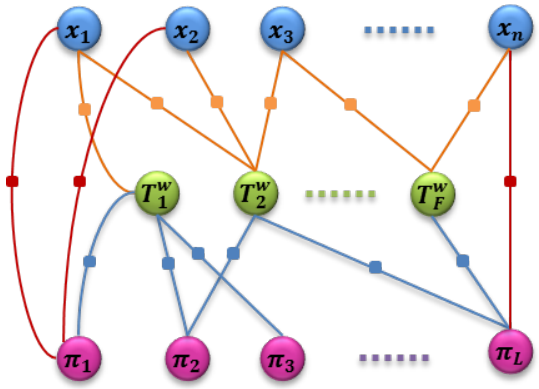


Figure 3: Sparse Point-Plane SLAM factor graph. The variable nodes include camera poses $\mathbf{T}_c^w$, points $\mathbf{x}_i$, and plane landmarks π_i. The factor nodes are point and plane measurements and also the constraints between points and planes on which they lie.

Note that the global bundle adjustment, when a loop closure is detected, is done in our point-plane SLAM using points and planes as landmarks. Loops are detected by ORB-SLAM2 which uses bag of words [Gálvez-López and Tardos, 2012] based on ORB features.

4 Experiments

To evaluate the performance of the proposed method, we use the publicly available TUM RGB-D dataset [Sturm *et al.*, 2012]. We show both qualitative and quantitative results for the proposed point-plane SLAM by comparing it with the state-of-the-art sparse mapping system, ORB-SLAM2 [Mur-Artal *et al.*, 2015] and also report the computational cost. Results are also reported for comparison against Pop-up SLAM [Yang *et al.*, 2016] in low-texture environments.

Sequences from the TUM dataset used in our experiments are given in Table 1 along with a short description of the type of scene observed in each sequence. These sequences have a wide range of conditions, from plane-rich scenes to scenes with little or no texture.

4.1 Qualitative Results

Qualitative results for five different sequences have been shown in Fig. 1 and Fig. 4. The first column shows the original ORB-SLAM2 frame along with the tracked

Dataset	Description
fr1/xyz	desk in an office environment
fr1/desk	sweeps over four desks in an office
fr1/floor	sweeps over wooden floor, planar
fr3/cabinet	little texture, planar surfaces
fr3/str_notex_near	low-texture close
fr3/str_notex_far	low-texture far
fr3/long_office	large loop around office desks

Table 1: Sequences used in the experiments

ORB features. The detected planes are shown in the second column and finally the last two columns depict the generated map consisting of points and planes from two different view-points. In the first row of Fig 4, we can see that the generated map is consistent with the ground truth since we can see the desk as a unique green plane representing the four same-height office desk in the sequence. The same scenario happens for the second sequence in the second row of Fig 4 where we have the red plane reconstructed from the floor which is observable in all of the frames of this dataset. The mapping and tracking results for two challenging low-texture datasets have been illustrated in Fig 4 third row and Fig 1 for `fr3/str_notex_near` and `fr3/str_notex_far` respectively. For these sequences, ORB-SLAM2 was unable to detect features in the environment with the default setting. The feature detection threshold had to be lowered to get some features which also resulted in more outlier and inaccurate trajectory. However, our point-plane SLAM system significantly improves the trajectory estimates and result in richer maps for these sequences.

In addition to the tracking accuracy in the frame-to-frame scenario, incorporating planes greatly impacts the trajectory estimate when there is a large loop closure: the system performs a global bundle adjustment with the points as well as plane landmarks. The produced map is more consistent as can be seen in the last row of Fig. 4, which is from the `fr3/long_office` sequence. The green desk and red floor along with the monitors in this map are consistent with the ground truth.

4.2 Quantitative Comparison

To evaluate the accuracy of the state estimation and implicitly the map reconstruction, we compare the proposed system with the traditional point-based ORB-SLAM2. Comparisons are reported in the terms of the following metrics: Absolute Trajectory Error (ATE), Relative Translational Error (RTE), and Relative Rotational Error (RRE). Results are given in Table. 2 for the sequences specified in Table. 1 that compare ORB-SLAM2 and our proposed point-plane SLAM.

It can be seen from the Table 2 that our point-plane SLAM improves the trajectory estimate in all of the sequences and up to 30% in some of them (the bold numbers). For instance, in the `fr1/desk` dataset, the Root-Mean-Squared-Error (RMSE) ATE has decreased significantly by 32.65%. The RTE and RRE has also decreased significantly which shows more accurate tracking and mapping. The reason for this improvement is the presence of four desks at the same height (all belonging to the same infinite plane), as shown in Fig 4 first row. These planar surfaces impose a strong structural constraint on where points can lie in the scene and our proposed method exploits these planar structures to improve the tracking and mapping of the scene. Another sequence in which we have a strong planar structure is `fr3/long_office`, where tables are connected to each other with monitors on top, as shown in Fig 4 last row. The presence of these planes particularly in the large loop closure present in this sequence improves the accuracy of tracking and mapping significantly, with a 29.93% reduction in ATE, 14.18% reduction in RTE, and 9.69% reduction in RRE.

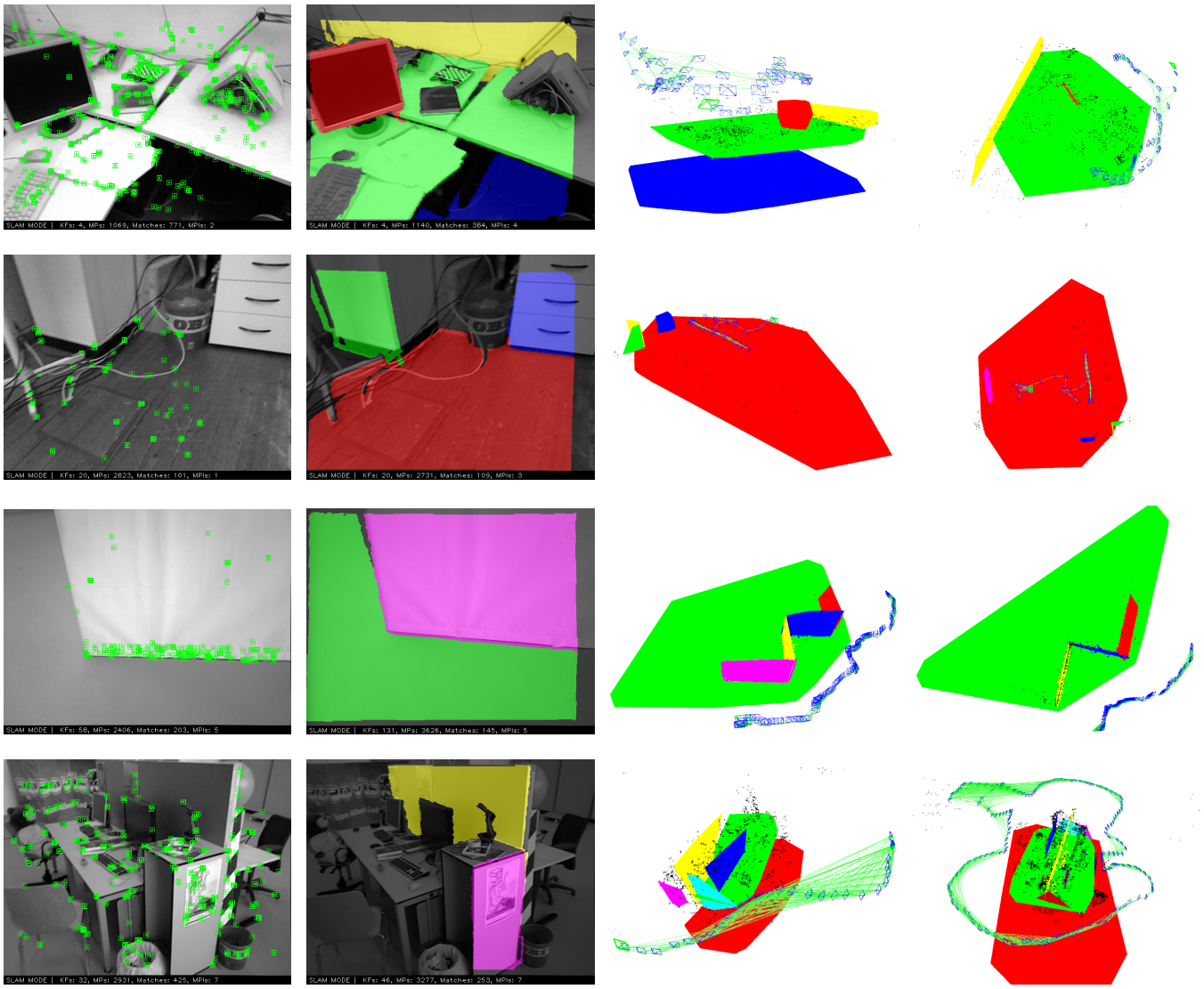
One of the most challenging scenarios for traditional ORB-SLAM2 are low-texture environments such as `fr3/str_notex_near` and `fr3/str_notex_far`. Our point-plane SLAM shows great improvement of 35.53% and 4.58% respectively in the ATE. The detected ORB features, in these sequences are on the intersections of planes. In these challenging low-texture environments, integrating planes in tracking and mapping shows its effectiveness demonstrably and yields richer and more semantically meaningful map (Fig 4 third row) rather that just sparse points belonging to the intersection of walls and floor.

We further compare our point-plane SLAM against the recent plane SLAM work, Pop-up SLAM [Yang *et al.*, 2016], which uses the point-based semi-dense LSD SLAM [Engel *et al.*, 2014] to provide photometric odometry constraint along a free unconstrained direction. The comparison is given in Table 3, the number for Pop-up SLAM are taken from [Yang *et al.*, 2016] for the `fr3/str_notex_far`. Even though Pop-up SLAM uses monocular input, the ground truth height of the camera is provided for initialisation of their system and the reported results are with absolute scale and hence comparable to our work that uses RGBD input.

As Table 3 suggests, our SLAM system outperforms Pop-up SLAM on this challenging sequence. Pop-up SLAM loses the track of small movements of the camera due to the lack of enough plane landmarks in this environment (under-constrained) and lack of frame-to-frame odometry. The generated map using the proposed system for this sequence can be seen in Fig. 1.

4.3 Analysis of the Runtime

All the experiments have been done on CPU Intel i7, 3.6 GHz and all of the code is implemented in C++.



(a) ORB_SLAM2 Frame (b) Detected Planes (c) Generated Map (Side) (d) Generated Map (Top)

Figure 4: Qualitative Results for 4 different TUM RGBD datasets.

Dataset	ATE (cm)		RTE (cm)		RRE (deg)	
	ORB-SLAM2	Ours	ORB-SLAM2	Ours	ORB-SLAM2	Ours
`fr1/xyz`	1.0457	0.9647	1.5693	1.4675	0.987095	0.953387
`fr1/desk`	2.2668	**1.5267**	4.0835	**3.2994**	1.854693	1.781697
`fr1/floor`	1.4399	1.3798	4.2161	3.8789	3.322874	2.885645
`fr3/cabinet`	7.9602	7.3724	1.51002	1.44081	6.863989	6.562287
`fr3/str_notex_near`	1.6882	**1.0883**	4.0383	**2.4540**	1.847615	1.154119
`fr3/str_notex_far`	2.0007	1.9092	3.4869	3.3523	0.847913	0.767912
`fr3/long_office`	1.5129	**1.0601**	3.0555	**2.6223**	0.892738	**0.806238**

Table 2: Comparison of RMSE for ATE, RTE, and RRE for 7 different TUM RGBD datasets. Numbers in bold represent over 30% improvement.

Dataset	Pop-up SLAM		Ours	
	Mean	Std	Mean	Std
`fr3/str_notex_far`	18 cm	7 cm	1.75 cm	0.76 cm

Table 3: Comparison of Mean and Std for ATE in the challenging low-texture `fr3/str_notex_far` dataset for Pop-up SLAM and our point-plane SLAM

The back-end optimizations have been implemented in the general framework for graph optimization, g2o [Kümmerle *et al.*, 2011]. The most time consuming part of the system is plane segmentation which takes on average 70 ms per frame (which can be reduced later by running in a multi-threaded pipeline). With the back-end optimisation, our system runs around 10Hz if we segment planes and track them in each frame. However we note that normally a plane in the scene can be easily tracked into the neighbouring frames so it is enough to detect and segment planes in each keyframe of the point-based ORB-SLAM2 instead of every frame. This would allow our system to run in near real-time at about 20Hz.

In Table 4 the runtime analysis of our SLAM system and average statistics of the evaluated RGBD datasets have been provided. All of theses timings have been measured when our system segment and match planes in each frame and the local mapping optimization is done with the presence of point and plane landmarks after adding each keyframe.

Main Components	Runtime
Plane segmentation	67.9 msec
Data association	2.6 msec
Local mapping optimization	12.4 msec
Total frame time	91.2 msec

Table 4: Sparse Point-Plane SLAM runtime average statistics on the evaluated RGBD TUM datasets with plane segmentation and tracking in each frame.

5 Conclusions

In this work, we explored the effects of incorporating plane as geometric entities in a sparse point-based SLAM framework. The improved performance due to using points and planes has been clearly shown by the experiments. Using planes is more effective in environment where there is dominant planar structure present. In cases where enough planes are not present, the point based SLAM can still function as usual. Currently, the method works with RGB-D input and plane segmentation is the bottleneck in terms of computation. In future, methods that can provide plane estimate from monocular input will be explored, which can then be incorporated in a purely monocular system. Also the further exploration will be introducing geometric and semantic relations between plane landmarks themselves.

References

[Cadena *et al.*, 2016] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[Engel *et al.*, 2014] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[Engel *et al.*, 2017] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[Forster *et al.*, 2014] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

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

[Gee and Mayol-Cuevas, 2006] Andrew P Gee and Walterio Mayol-Cuevas. Real-time model-based slam using line segments. In *International Symposium on Visual Computing*, pages 354–363. Springer, 2006.

[Gomez-Ojeda *et al.*, 2017] Ruben Gomez-Ojeda, Francisco-Angel Moreno, Davide Scaramuzza, and Javier Gonzalez-Jimenez. Pl-slam: a stereo slam system through the combination of points and line segments. *arXiv preprint arXiv:1705.09479*, 2017.

[Grisetti *et al.*, 2010] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[Hartley and Zisserman, 2003] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[Holzer *et al.*, 2012] Stefan Holzer, Radu Bogdan Rusu, M Dixon, Suat Gedikli, and Nassir Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE, 2012.

[Kaess, 2015] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4605–4611. IEEE, 2015.

[Kümmerle *et al.*, 2011] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[Lemaire and Lacroix, 2007] Thomas Lemaire and Simon Lacroix. Monocular-vision based slam using line segments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2791–2796. IEEE, 2007.

[Mur-Artal *et al.*, 2015] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[Newcombe *et al.*, 2011a] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[Newcombe *et al.*, 2011b] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[Prisacariu *et al.*, 2017] Victor Adrian Prisacariu, Olaf Kähler, Stuart Golodetz, Michael Sapienza, Tommaso Cavallari, Philip HS Torr, and David W Murray. Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*, 2017.

[Pumarola *et al.*, 2017] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francesc Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *Proc. International Conference on Robotics and Automation (ICRA), IEEE*, 2017.

[Sturm *et al.*, 2012] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[Taguchi *et al.*, 2013] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5182–5189. IEEE, 2013.

[Trevor *et al.*, 2013] A Trevor, Suat Gedikli, R Rusu, and H Christensen. Efficient organized point cloud segmentation with connected components. In *3rd Workshop on Semantic Perception Mapping and Exploration (SPME), Karlsruhe, Germany*, 2013.

[Yang *et al.*, 2016] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. Pop-up slam: Semantic monocular plane slam for low-texture environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1222–1229. IEEE, 2016.