Structure-aware SLAM with Planes in Man-Made Environment

Jiyuan Zhang[⋆], Gang Zeng, and Hongbin Zha

Key Laboratory on Machine Perception, Peking University, Beijing, China
{zhangjiyuan.eecs,zeng}@pku.edu.cn,zha@cis.pku.edu.cn

Abstract. We propose to utilize co-planar constraints in point-based simultaneous localization and mapping of man-made environment. Planes are common structures in both indoor and outdoor city scenes. They are good features in two ways: strong constraint on points, and suitable supplementary for long-distance tracking. Large structural planes in optimization-based framework can reduce drifting error by connecting many points and cameras. Our method can detect and locate planes from multiple images and sparse 3D points. No dense reconstruction or triangulation is needed. The planes are actively extended as SLAM system goes on, making our algorithm suitable for exploration-style applications. The proposed method is tested on real world indoor and outdoor long video sequences, showing the capability to significantly reduce drifting error.

Keywords: SLAM, plane detection, vanishing points, color segmentation, non-linear optimization

1 Introduction

Simultaneous Localization And Mapping (SLAM) is an important task in computer vision. It is widely used in applications such as robotics, auto driving and 3D reconstruction. Feature-based visual SLAM works [18, 8, 13] use feature points as input, 3D points as landmark. There are also direct methods [6, 3, 4] that process every pixel in image. The keypoints or pixels are always treated as independent elements, either to reduce computation cost or to be robust to errors. Only when dense reconstruction is needed [19–21] some normalization is imposed on pixels, which increases complexity greatly.

However more information can be found in images. It is necessary to utilize relation of points in visual SLAM to achieve better results. In this paper we consider general man-made environments, in which a great variety of additional structural information can be used. A SLAM system with ability to recognize and utilize these knowledge would perform better. In the proposed method we detect

[⋆] Corresponding Author. Student. Room 2207, Science Building No.2, Key Laboratory on Machine Perception, Peking University, Beijing, China. 100871. Tel: +86-010-62757069.

and locate structural 3D planes from image and sparse 3D points, and use them to help improve the SLAM result. Planes can be tracked longer than individual feature points, especially in visual exploration tasks, as shown in Fig. 1. Even if there is no co-visible feature point, the structure still provides the constraint to all frames seeing it. By employing planes, the commonly seen drifting error could be reduced, though never completely eliminated unless explicit loop closing is made.

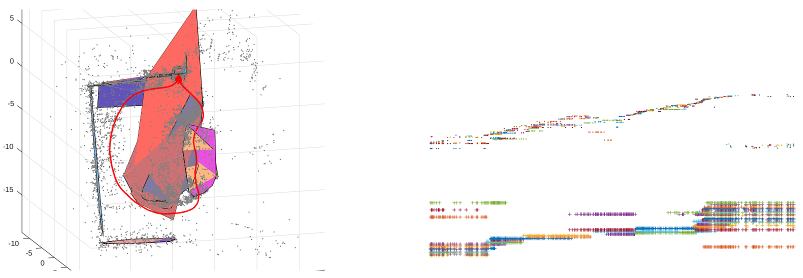


Fig. 1. Left: 3D points and planes of an outdoor scene, viewed from top. Right: the visibility of keypoints (top half, dot) and planes (bottom half, cross) through time (x-axis). The planes are much better for long-term tracking in man-made environments.

Different from points or lines, there is no "projected image" for an ideal infinite plane. It has to be reconstructed from some 3D source. A common choice is RANSAC in point cloud. Random sampling could be problematic if the cloud is sparse and not evenly distributed, which is usually the case for feature-based point landmarks. Even with a well-developed SLAM system, reconstructed sparse 3D points can only give basic impression of scene structures. Human viewers can understand the scene by adopting a lot of prior knowledge available in the input images, which is in our interest.

Our method detects potential co-planar points in a set of images, with the aid of sparse reconstruction of 3D points. First the images are over-segmented by color, as single-color regions are more likely to be planar. Such regions provide single-view co-planar information. Gathered over multiple views, the relation is more stable and ready to be clustered into co-planar groups. Then vanishing points in images are used to limit the search space of planes. The two information sources are combined to detect planes, after which structure constraints are used to associated points. The planes, points and camera poses are then jointly optimized to give consistent result.

2 Related Work

Many works have explored the idea of using structures in SLAM. Information can be as high level as objects [23], or as simple as planes or lines. It is easier to extract geometry structure from depth sensors, like RGB-D camera [10, 24, 11]. For monocular images, extra effort must be paid. Planar regions of texture could be tracked through homography [12, 15], providing constraints for motion. However to build a meaningful and stable error function, texture must be compared, leading to a dense or semi-dense color consistency check. Texture-less planar regions have to be matched by shape. DPPTAM [3] detects and maintains such regions in 3D, but only use them for dense reconstruction. The planes do not participate in motion estimation. If indoor scenes with floor and walls are assumed, pop-up structures [28] can be detected for low-texture regions.

Pure plane-based work [29] assumes one or more large planes are present in the scene, and detects them by applying RANSAC iteratively. The planes play the major role in motion computation in the form of homography. However it is an off-line framework, batching all images together.

A recent work [22] also exploit the indoor planes in the dense SLAM system. All pixels are reconstructed as surfels, then divided to planar and non-planar types. Planes are made from combined surfels as memory-efficient representation. It is implemented in the manner of depth fusion for dense reconstruction. Our work is different in that discrete points are still kept. Planes are not employed to fill the map, but only impose constraints to points. Point-on-plane condition is represented by re-projection error, which is view-dependent. This flexibility can tolerate unstable points commonly seen in outdoor scenes, where features can be very far from camera. Also we allow planes to cross color boundary to establish long-term landmarks, helping large-scale visual odometry, while dense depth map of [22] is more suitable for small-scale indoor SLAM.

The Multilayer Feature Graph (MFG) [16] is proposed to combine multiple types of features, including point, line segment, vanishing point and plane. After point and line segment features are extracted from image, other color information is dropped. The planes are detected with RANSAC among spatial points and lines, using 3D distance measurement. All the features are detected and associated with others by various kinds of relationship. Then an overall non-linear optimization is performed to solve both structure and motion.

Previous works [2, 9] using planes as features, detected directly from sparse point cloud with RANSAC. It is applicable for the small-scale AR scene, but not for larger outdoor scenes where point cloud is too sparse. Later [17] on the other hand uses point with normal to represent planes. This allows planes and points to be tracked in a unified framework of Extended Kalman Filter. The connections between co-planar points are built simply by geometry and not used in filter update.

3 Planes in Local Map

Our work is based on keypoint-based monocular SLAM framework. Sparse 3D points $\{\mathbf{X}\}$ are reconstructed first. Large-scale loop closure is not employed, so that long-term drifting will be significant. Our work focus on the newest part of the map, called the local map. It originates from the newest keyframe and all 3D points $\mathbf{X}_i$ visible in it. Then all keyframes in which any $\mathbf{X}_i$ is visible, and any other 3D points visible in these keyframes, are also included. Limited by scene co-visibility and memory issue, the number of keyframes in local map seldom reaches the scale of 100.

Planes are detected and located in the local map with points assigned. Details will be given in the rest of this section. Both points and planes are put into the structure-and-pose bundle adjustment over the local map. Let $\pi(\mathbf{X})$ be the projection from 3D to 2D, and $\mathbf{u}$ the observed feature point. In addition to conventional point re-projection error

$$e_{proj}(\mathbf{X}, \mathbf{u}) = |\pi(T_{cw} \cdot \mathbf{X}) - \mathbf{u}| \ , \tag{1}$$

the planes introduces two new kinds of edge:

Point-on-plane projection For a plane $\mathbf{p} = (\mathbf{n}^T, d)^T$ and a point $\mathbf{X}$ assigned with $\mathbf{p}$, the closest point

$$\mathbf{X}_{\mathbf{p}} = \mathbf{X} - (\mathbf{n}^T \mathbf{X} + d)\mathbf{n} \tag{2}$$

exactly on $\mathbf{p}$ is considered the additional source of projection.

$$e_{plane\_proj}(\mathbf{X}, \mathbf{p}, \mathbf{u}) = e_{proj}(\mathbf{X}_{\mathbf{p}}, \mathbf{u}) \tag{3}$$

is also included for all observations $\mathbf{u}$. This error measurement imposes less weight if $\mathbf{p}$ is frontal-parallel in all views, avoiding additional view-dependent weights. Any criteria of point re-projection errors could be shared.

Spatial point to plane Non-local 3D points not visible in local map may also be included in large planes. Those points are fixed during the optimization as anchors for plane. Point-plane distance

$$e_{space}(\mathbf{X}_{old}, \mathbf{p}) = |\mathbf{n}^T \mathbf{X}_{old} + d| \tag{4}$$

is used to describe non-local points on plane relationship. This distance is included so that planes can impose constraints across longer range, both spatial and temporal.

The final objective function

$$E = \sum_{\mathbf{X}} \left[e_{proj}(\mathbf{X}, \mathbf{u})^2 + \sum_{\mathbf{p}} e_{plane\_proj}(\mathbf{X}, \mathbf{p}, \mathbf{u})^2 \right] + \lambda \sum e_{space}(\mathbf{X}_{old}, \mathbf{p})^2 \tag{5}$$

is then minimized with Gauss–Newton algorithm. The parameter λ is employed to balance different units. e_{space} is measured in arbitrarily scaled spatial distance while e_{proj} in pixel. λ should be chosen according to the scene, with tolerances of spatial error σ_{space} and pixel error σ_{pixel}. σ_{space} is also used in the creation of plane candidates later in §3.3.

As planes are included, the objective function is different from conventional point-based Bundle Adjustment (BA). The error term $e_{plane\_proj}(\mathbf{X}, \mathbf{p}, \mathbf{u})$ modified the structure of BA by introducing terms with three parameter blocks.

However by carefully sorting the blocks and using the fact that large structural planes are much fewer than points, it is still possible to solve the Jacobian equation with Schur complement in reasonable time. Timing results will be discussed in §4.1.

3.1 Single-Color Regions

It is assumed that single-color regions in image are likely to be planar, which is applicable in many man-made environments. We only need such assumption to find connections between features, which are lost from the very beginning of feature point extraction. It is possible to drop any advanced single-view plane detection algorithm here without major change of the rest of our algorithm.

Graph-based method [7] is used to create over-segmentations on input image, example shown in left of Fig. 2. Only large regions are considered. The points in the same region are more likely to be coplanar. As features are often close to color boundaries, the regions are dilated by a few pixels before deciding membership of features in this image. Then one feature can be assigned to multiple regions, which forges connection across color boundary to provide structural constraints at larger scale. It is common that a real large plane consists of several single-color regions.

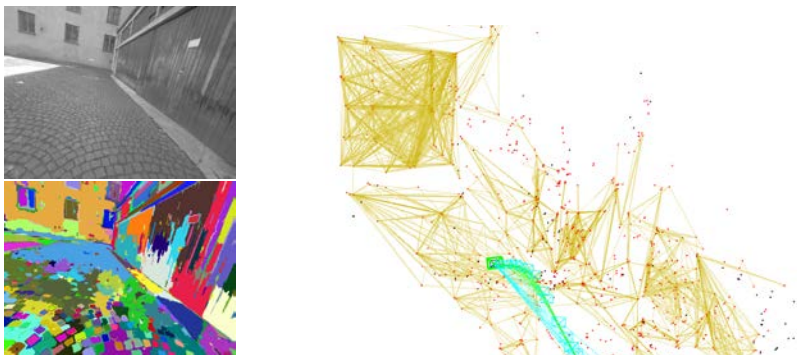


Fig. 2. Left top: input keyframe. Left bottom: color segmentation. Right: region-based similarity of 3D points, connected if seen in enough single-color regions. Best viewed in color.

The region-based similarity $\mathbf{S}(i,j)$ is defined as the number of regions in all local keyframes where 3D points $\mathbf{X}_i$ and $\mathbf{X}_j$ are both seen inside. Note this number might be larger than simply counting same-region keyframes, because one point could be covered by several *dilated* regions. The number of regions is used, instead of keyframes, to emphasis the similarity of feature pairs close to the same color boundary. Whether there is depth discontinuity or not, the color boundary itself is likely to be planar in man-made environment.

As shown in the right of Fig. 2, the accumulated **S** roughly sketches three major planes in the scene, with a few outliers.

3.2 Line-based Structures

Straight lines are another type of common and important feature in man-made environment, being good hints for presence of planes. Parallel line groups, or Vanishing Points (VP) are used in this work to limit the normal of plane candidates.

First the lines are detected with LSD [27]. Vanishing points are clustered from each frame using J-Linkage [25]. Each VP in image represents a direction $\mathbf{v}_c$ in camera coordinates. With known camera-to-world transformation T_{wc}, the VP directions are unified in the world coordinate as $\mathbf{v}_w = R_{wc}\mathbf{v}_c$. Though not matched by image textures, observations of same VP are close in space. As frames coming in, the significant VPs are seen again and again, while secondary or erroneous directions can be simply rejected by counting.

The potential normals are computed from cross product of VPs and cleaned by merging close ones. One plane normal is supported by a VP if they are orthogonal. We use strong criteria that valid plane normals should be supported by at least two VP directions, so only major structural planes are detected. Spatial sampling of plane normal is no longer needed. Fig. 3 shows the normals computed from accumulated significant VP directions.

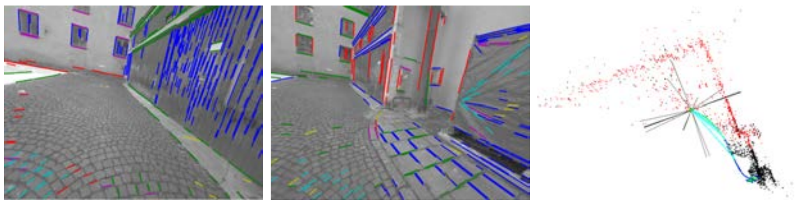


Fig. 3. Left and middle: examples of vanishing point groups on two keyframes. Note the VP groups are not matched between images. Right: top view of potential normals in local map, computed from accumulated vanishing point directions. Best viewed in color.

3.3 Plane Candidates

We construct planes in the local map from two different sources, both using restricted plane normals from VP directions. We use MLESAC [26] to detect planes from a set of 3D points. The measurement, point-plane distance $e_{space}(\mathbf{X}, \mathbf{p})$, is easily computed given the plane normal. Only one point is sampled in each iteration. The error threshold σ_{space} is chosen according to the scale of the scene.

One way is from single-color regions. For each $\mathbf{X}_i$ one plane is detected with $\{\mathbf{X}_j | \mathbf{S}(i,j) > t_{color}\}$, with t_{color} a fixed threshold. Overlapping among clusters are allowed, so that one point may be assigned to several planes.

The other way is direct MLESAC using all 3D points in the local map. The required amount of inliers are proportional with that of local 3D points, much more than region-based one. This method is kept to recover planes with good texture, on which many keypoints but no large color regions are detected.

Many of these planes are duplicates from different sources, which is also a kind of similarity. The planes are reversely indexed by inlier points, thus each point $\mathbf{X}_i$ can have zero or more potential planes $\mathcal{P}_i$ populated from both sources. The points are then clustered again with similarity $\mathbf{S}$, requiring not only $\mathbf{S}(i,j) > t_{local}$ but also $\mathcal{P}_i \cap \mathcal{P}_j \neq \emptyset$. We choose $t_{local} < t_{color}$ to encourage large planes. Detail of this greedy clustering is described in Algorithm 1. Note this is different from the previous clustering by allowing center of cluster changed during the expansion. The clustered planes are then compared with previous structural ones, and merged if very close.

```
Data: available anchor 3D points $A \subseteq \{\mathbf{X}_i\}$, potential planes for each point
      $\{\mathcal{P}_i\}$, region-based similarity $\mathbf{S}$, potential normals $\{\mathbf{n}\}$, threshold $t_{local}$
Result: clusters of points $\{C\}$ and planes $\{\mathbf{p}\}$
compute degrees of each $\mathbf{X}_i$: $D_i = \sum_{j \neq i} \mathbf{S}(i,j)$;
while $A \neq \emptyset$ do
    pick anchor $\mathbf{X}_a \in A$ with largest $D_a$;
    let $C = \{\mathbf{X}_a\} \cup \{\mathbf{X}_j | \mathbf{S}(a,j) > t_{local}\}$;
    fit best plane $\mathbf{p}$ with $C$ and $\{\mathbf{n}\}$;
    let potential planes $\mathcal{P} = \{\mathbf{p}\}$;
    repeat
        for $\mathbf{X}_c \in C$ do
            let $J = \{j | \mathbf{S}(c,j) > t_{local}, \mathcal{P}_j \cap \mathcal{P} \neq \emptyset \vee \mathcal{P}_j = \emptyset\}$;
            let $C = C \cup \{\mathbf{X}_j | j \in J\}$;
            let $\mathcal{P} = \mathcal{P} \cup \bigcup_{j \in J} \mathcal{P}_j$;
        end
        fit best plane $\mathbf{p}$ with $C$ and $\{\mathbf{n}\}$, remove outliers from $C$;
    until $C$ unchanged;
    remove $C$ from $A$;
end
```

Algorithm 1: Greedy clustering with color information and potential planes of each point. $|\mathcal{P}| \geqslant 1$ for each cluster.

To speed up the second clustering, the previous planes are extended first. $\mathbf{X}_i$ in local map is assigned to a previous plane $\mathbf{p}_k$ if $e_{space}(\mathbf{X}_i, \mathbf{p}_k)$ is small enough *and* there is some $\mathbf{X}_j$ assigned to $\mathbf{p}_k$ satisfies $\mathbf{S}(i,j) > t_{local}$. The occupied 3D points are not used as starting of clustering, but still can be added to other clusters.

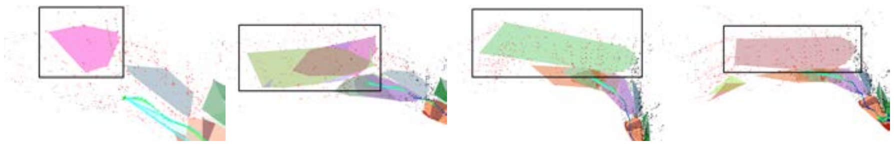


Fig. 4. The detected planes in local map near Fig. 3, timestamps from left to right. Note how the farthest plane (marked by black box) is growing in size by assigning points to it. Similar planes are actively merged.

4 Experiments

Our experimenting system is built based on ORB-SLAM [18], with explicit loop-closure disabled. Note our method does not perform loop-closure with planes. Previously tracked planes are expanded only if they share some regions in the local map, which requires continuous visibility. The process of lines and planes is inserted after local map updating, refining both 3D points and camera poses. Plane detection may fail if too few are found in single-color regions. In this case, only points are refined and ORB-SLAM goes on.

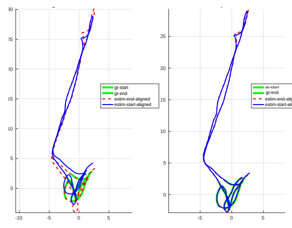
Table 1. Errors of ORB-SLAM and proposed method on sequences of dataset [5]. Bold indicates measurements *not* improved by our method.

		ORB-SLAM				proposed			
		E_{align}	E_R	E_S	RMSE	E_{align}	E_R	E_S	RMSE
Indoor	Seq. 16	0.9709	0.6469	0.9240	0.5017	0.1792	0.3237	0.9847	0.0498
	Seq. 28	8.0150	**18.5386**	0.5773	**0.2868**	5.0024	19.5471	0.8470	0.2974
	Seq. 35	5.1530	3.4454	0.7500	0.5830	3.1686	2.8149	0.7988	0.4849
	Seq. 36	2.2379	2.9817	0.8411	0.5757	1.1187	2.1175	0.9942	0.3841
Outdoor	Seq. 23	17.6954	**1.6574**	0.3709	0.4455	3.6029	2.2916	0.7808	0.4357
	Seq. 30	2.6210	1.1645	0.8409	0.6261	1.3844	0.9717	1.0845	0.2267
	Seq. 45	2.8350	2.2240	0.8240	0.1788	1.5819	1.7018	0.9000	0.1785

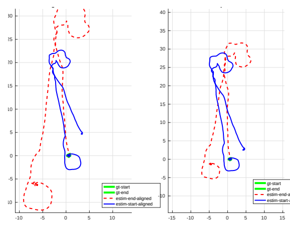
Our method is tested on sequences of TUM monocular visual odometry dataset [5] with both indoor and outdoor man-made scenes. The dataset is created for evaluation of visual odometry systems. Every sequence are closed loop, with the same well-textured objects at the both ends where ground-truth of motion is provided. We follow the evaluation method of the authors by comparing the drifting of the loop. The recovered trajectory is aligned by scaled transformation $Sim(3)$ twice, with the ground-truth at the beginning and the end. Fig. 5 shows some trajectories of ORB-SLAM (left) and ours (right). Two aligning transformations T_A and T_E are computed, and $T_{align} = T_A^{-1} T_E$. The alignment error measurement E_{align} is the average positional displacement of two aligned trajectories. Other useful error measurements are:

- E_R: rotation part of T_{align}, in degrees;
- E_S: scale part of T_{align}, should be close to 1;
- RMSE: average positional drifting using a single alignment, only partially available.

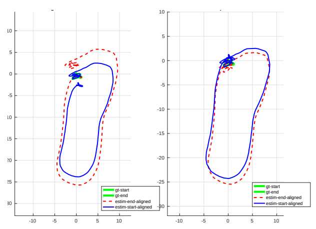
Comparison of error values is shown in Table 1.



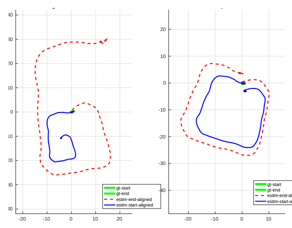
(a) Seq. 16



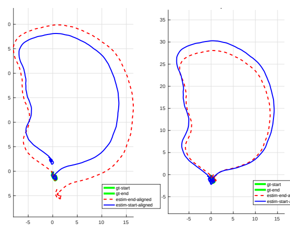
(b) Seq. 28



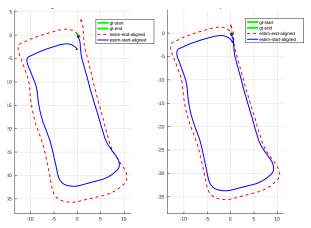
(c) Seq. 36



(d) Seq. 23



(e) Seq. 30

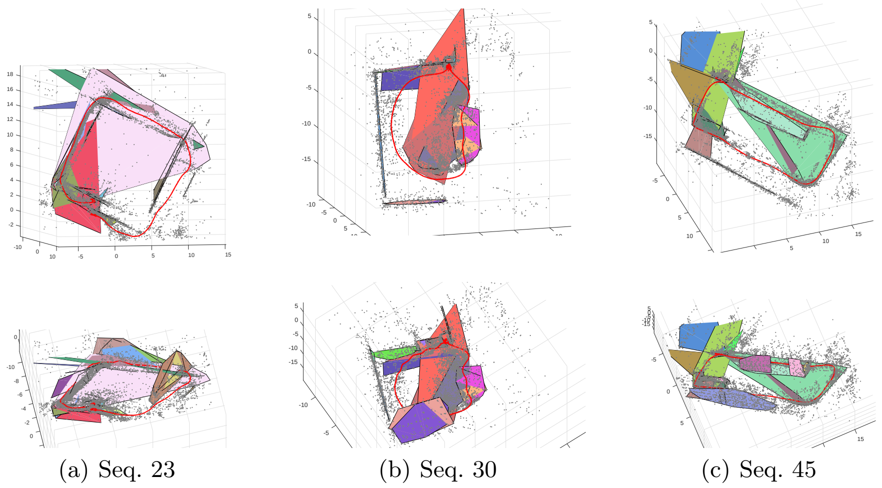


(f) Seq. 45

Fig. 5. Motion from indoor (top row) and outdoor (bottom row) sequences. For each sequence, ORB-SLAM at left and our result at right are aligned to ground truth at the beginning (blue, solid) and the end (red, dashed). The closer two trajectories, the better result. See experiment section for detail.

Most indoor sequences are captured in narrow corridors and medium-sized offices, sometimes with going up or down stairs. The structural planes mainly comes from walls and floors, with noncontinuous texture. The major rotation error shown in Fig. 5(b) is introduced from climbing up spiral stairs, where walls are texture-less but circular. The significant drifting of ORB-SLAM on the stairs is not corrected with plane assumption. However the scale is correctly kept.

In the outdoor scenes the structural planes come from ground and facades. As we allow shared points between detected planes, the ground plane can grow very large even if texture changes. Long-tracked planes can improve quality of motion by forcing consistent scale, as shown in Fig. 5(d). Some of recovered structure planes are shown in Fig. 6 as convex hull of footprints of associated 3D points, filled with random color. Major planes in the scene are successfully detected and reconstructed. Most of them are real ones with texture, while some are virtual. Virtual planes do not harm SLAM results, as they provide same



(a) Seq. 23 (b) Seq. 30 (c) Seq. 45

Fig. 6. Structural planes reconstructed for some outdoor scenes, top view and bird view.

level of structural constraints on 3D points. It is easy to reject virtual ones by checking color consistency if textured reconstruction is necessary.

4.1 Timing Issues of Mixed Objective Function

It is necessary to finish the optimization in time for a real-time SLAM application. The work of Sparse Bundle Adjustment (SBA) [14] has shown the conventional point-and-camera objective function with re-projection error could be effectively minimized using Schur complement. The key of Schur complement is the block diagonal pattern of $J^T J$ when solving the Jacobian linear system $J\delta\mathbf{x} = \epsilon$. In typical BA problem, most degrees of freedom are occupied by 3D points. There is only one point in each error term, so that the columns of Jacobian matrix J corresponding to the points are mutually orthogonal, producing a large block-diagonal sub-matrix in $J^T J$. It is much easier to invert this sub-matrix, leading to faster block Gaussian elimination.

The introduction of planes in the objective function changes the pattern of J. However the key part, block-diagonal sub-matrix in $J^T J$ corresponding with 3D points, still exists. The variables of points can still be eliminated with Schur complement. The rest includes planes additionally, more than just cameras in conventional BA, thus it will be slower. But large structural plane is not that many. Their impact on performance should not exceed equal number of additional cameras.

To demonstrate the time cost of optimization by comparing the following 3 objective functions:

– *full*: the proposed one in Eqn. 5,

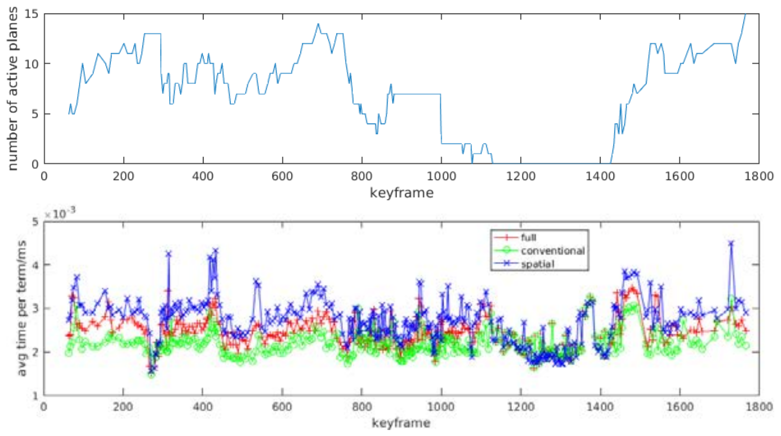


Fig. 7. Top: number of planes in the local map of Seq. 30. Bottom: time cost per term in one iteration, in milliseconds. There is no active plane around Frame 1200, where *full* falls back to *conventional*.

- *conventional*: only $e_{proj}(\mathbf{X}, \mathbf{u})$ are included, planes ignored,
- *spatial*: use e_{space} instead of e_{proj} regardless of the point $\mathbf{X}$ is fixed or not.

Because the total number of terms $N(e)$ are different in these objective functions, the time of each iteration is divided by $N(e)$ accordingly. The optimization is performed with Google Ceres Solver [1] in C++, for maximal 5 iterations.

Table 2. Average statistics of objective function optimization. $N(e)$ is the number of that type of error term. (*conven.* short for *conventional*)

		$N(e_{proj}(\mathbf{X_p}))$	$N(e_{space})$	frame time (ms)	time per term (ms)
Seq. 23	*full*	7222	1393	886.5194	0.0024
$N(e_{proj}(\mathbf{X}))$	*conven.*	-	-	662.8082	0.0022
$= 48587$	*spatial*	-	3089	933.2013	0.0028
Seq. 30	*full*	2406	323	657.9031	0.0024
$N(e_{proj}(\mathbf{X}))$	*conven.*	-	-	548.6552	0.0022
$= 40794$	*spatial*	-	641	711.7544	0.0027
Seq. 45	*full*	1663	425	506.9243	0.0023
$N(e_{proj}(\mathbf{X}))$	*conven.*	-	-	420.9805	0.0021
$= 32739$	*spatial*	-	692	555.9996	0.0026

Fig. 7 shows the number of planes and the per-term time cost in one iteration for Seq. 30. The number of active planes are kept less than 20 in this outdoor scene. The average statistics for some sequences are shown in Table 2. The major part, conventional re-projection error terms $e_{proj}(\mathbf{X}, \mathbf{u})$, are the same among three objective functions. *full* introduces extra terms with planes. There is visible performance impact that time cost per term is increased. The overall time for a local map optimization is increased but still applicable in the ORB-SLAM framework. It is possible to replace the conventional local BA with proposed objective function without major rework of a multi-threaded real-time SLAM system.

Note that the *spatial* objective function introduces less extra terms, but costs more time to optimize.

5 Conclusion

In this paper we propose a structure-aware SLAM method utilizing lines and planes in man-made environment. With clustered vanishing points and color-based image segmentation, planar structures are detected from images and sparse 3D points. Using more information from image improves visual SLAM result. The ability to track and update planes for long range reduces the drifting error of camera trajectory. The detected planes can also be used as base of large-scale dense reconstruction.

Acknowledgments This work is supported by National Natural Science Foundation of China (NSFC) 61375022 and 61403005.

References

1. Agarwal, S., Mierle, K., Others: Ceres solver. `http://ceres-solver.org`
2. Chekhlov, D., Gee, A.P., Calway, A., Mayolcuevas, W.W.: Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. International Symposium on Mixed and Augmented Reality pp. 153–156 (2007)
3. Concha, A., Civera, J.: Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. Intelligent Robots and Systems pp. 5686–5693 (2015)
4. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. In: arXiv:1607.02565 (July 2016)
5. Engel, J., Usenko, V., Cremers, D.: A photometrically calibrated benchmark for monocular visual odometry. In: arXiv:1607.02555 (July 2016)
6. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM, pp. 834–849. Springer International Publishing, Cham (2014)
7. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)
8. Forster, C., Pizzoli, M., Scaramuzza, D.: Svo: Fast semi-direct monocular visual odometry. International Conference on Robotics and Automation pp. 15–22 (2014)
9. Gee, A.P., Chekhlov, D., Mayolcuevas, W.W., Calway, A.: Discovering planes and collapsing the state space in visual slam. British Machine Vision Conference (2007)
10. Hsiao, M., Westman, E., Zhang, G., Kaess, M.: Keyframe-based dense planar SLAM. In: IEEE Intl. Conf. on Robotics and Automation, ICRA. Singapore (May 2017), to appear

11. Kaess, M.: Simultaneous localization and mapping with infinite planes. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 4605–4611 (May 2015)
12. Kahler, O., Denzler, J.: Implicit feedback between reconstruction and tracking in a combined optimization approach. DAGM Symposium (2008)
13. Klein, G., Murray, D.W.: Parallel tracking and mapping for small ar workspaces. International Symposium on Mixed and Augmented Reality pp. 225–234 (2007)
14. Lourakis, M.A., Argyros, A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Trans. Math. Software 36(1), 1–30 (2009)
15. Lovegrove, S., Davison, A.J., Ibãnez-Guzmán, J.: Accurate visual odometry from a rear parking camera. IEEE Intelligent Vehicles Symposium (June 2011)
16. Lu, Y., Song, D.: Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. IEEE Transactions on Robotics 31(3), 736–749 (June 2015)
17. Martinezcarranza, J., Calway, A.: Unifying planar and point mapping in monocular slam. British Machine Vision Conference (2010)
18. Murartal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics 31(5), 1147–1163 (2015)
19. Newcombe, R., Lovegrove, S., Davison, A.J.: Dtam: Dense tracking and mapping in real-time. International Conference on Computer Vision pp. 2320–2327 (2011)
20. Pizzoli, M., Forster, C., Scaramuzza, D.: Remode: Probabilistic, monocular dense reconstruction in real time. International Conference on Robotics and Automation pp. 2609–2616 (2014)
21. Ranftl, R., Vineet, V., Chen, Q., Koltun, V.: Dense monocular depth estimation in complex dynamic scenes. Computer Vision and Pattern Recognition pp. 4058–4066 (2016)
22. Salasmoreno, R.F., Glocken, B., Kelly, P.H.J., Davison, A.J.: Dense planar slam. International Symposium on Mixed and Augmented Reality pp. 157–164 (2014)
23. Salasmoreno, R.F., Newcombe, R., Strasdat, H., Kelly, P.H.J., Davison, A.J.: Slam++: Simultaneous localisation and mapping at the level of objects. Computer Vision and Pattern Recognition pp. 1352–1359 (2013)
24. Taguchi, Y., Jian, Y., Ramalingam, S., Feng, C.: Point-plane slam for hand-held 3d sensors. International Conference on Robotics and Automation pp. 5182–5189 (2013)
25. Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. European Conference on Computer Vision pp. 537–547 (2008)
26. Torr, P.H.S., Zisserman, A.: Mlesac: a new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78(1), 138–156 (2000)
27. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(4), 722–732 (2010)
28. Yang, S., Song, Y., Kaess, M., Scherer, S.: Pop-up slam: Semantic monocular plane slam for low-texture environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (October 2016)
29. Zhou, Z., Jin, H., Ma, Y.: Robust plane-based structure from motion. Computer Vision and Pattern Recognition pp. 1482–1489 (2012)