# Pop-up SLAM: Semantic Monocular Plane SLAM for Low-texture Environments

Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer

*Abstract*—Existing simultaneous localization and mapping (SLAM) algorithms are not robust in challenging low-texture environments because there are only few salient features. The resulting sparse or semi-dense map also conveys little information for motion planning. Though some work utilize plane or scene layout for dense map regularization, they require decent state estimation from other sources. In this paper, we propose real-time monocular plane SLAM to demonstrate that scene understanding could improve both state estimation and dense mapping especially in low-texture environments. The plane measurements come from a pop-up 3D plane model applied to each single image. We also combine planes with point based SLAM to improve robustness. On a public TUM dataset, our algorithm generates a dense semantic 3D model with pixel depth error of 6.2 cm while existing SLAM algorithms fail. On a 60 m long dataset with loops, our method creates a much better 3D model with state estimation error of 0.67%.

## I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is widely used for tasks including autonomous navigation, 3D mapping and inspection. Various sensors can be used for SLAM such as laser-range finders cameras, and RGB-D depth cameras. Monocular cameras are a popular choice of sensor on robots as they can provide rich visual information at a small size and low cost. They are especially suitable for weight constrained micro aerial vehicles that can carry only one camera. Therefore, in this work we focus on using monocular images to estimate the pose and map of the environment.

On one hand, many existing visual SLAM methods utilize point features such as direct LSD SLAM [1] and feature based ORB SLAM [2]. These methods track features or high-gradient pixels across frames to find correspondences and triangulate depth. They usually perform well in environments with rich features but cannot work well in low-texture scenes as often found in corridors. In addition, the map is usually sparse or semi-dense, which does not convey much information for motion planning.

On the other hand, humans can understand the layout, estimate depth and detect obstacles from a single image. Many methods have been proposed to exploit the geometry cues and scene assumption in order to build a simplified 3D model. Especially in recent years, with the advent of Convolutional Neural Networks (CNN) [3], performance of visual understanding has been greatly increased.

In this paper, we combine scene understanding with traditional v-SLAM to increase the performance of both

The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA. {shichaoy, songyu, kaess, basti}@andrew.cmu.edu
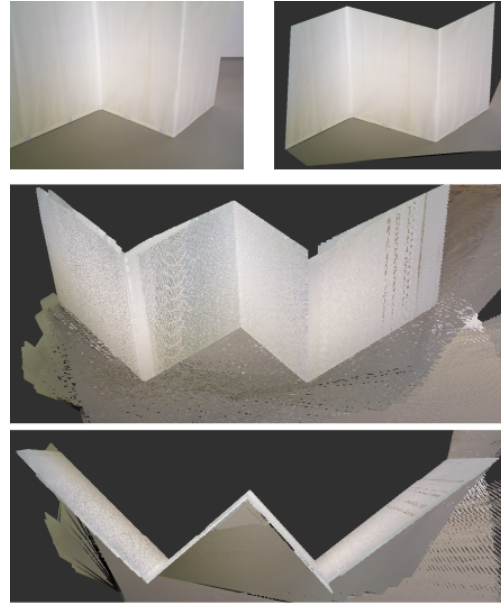
Fig. 1. 3D reconstruction on low-texture TUM dataset. (top) Single frame, raw image on left and 3D pop-up plane model on right. (center) Dense reconstruction using our *Pop-up Plane SLAM*. Each plane has a label of either a specific wall or ground. (bottom) Top view of the 3D model. Existing state-of-art SLAM algorithms fail.

state estimation and dense mapping especially in low-texture environments. We use a single image pop-up plane model [4] to generate plane landmark measurements in SLAM. With proper plane association and loop closing, we are able to jointly optimize scene layout and poses of multiple frames in the SLAM framework. In the low-texture environment of Figure 1, our algorithm can still generate dense 3D models and decent state estimates while other state-of-the-art algorithms fail. However, plane SLAM can easily be under-constrained, hence we propose to combine it with traditional point-based LSD SLAM [1] to increase robustness.

In summary, our main contributions are:

- A real-time monocular plane SLAM system incorporating scene layout understanding,
- Integrate planes with point-based SLAM for robustness,
- Outperform existing methods especially in some low-texture environments and demonstrate the practicability on several large datasets with loops.

In the following section, we discuss related work. Section III describes the single image layout understanding, which provides plane measurements for plane SLAM. In Section IV, we introduce the *Pop-up Plane SLAM* formulation and

combine it with LSD SLAM in Section V. Experiments on a public TUM dataset and actual indoor environments are presented in Section VI. Finally, we conclude in Section VII.

## II. RELATED WORK

Our approach combines aspects of two research areas: single image scene understanding and multiple images visual SLAM. We provide a brief overview of these two area.

### A. Single Image

There are many methods that attempt to model the world from a single image. Two representative examples are cuboidal room box model proposed based on vanishing point by Hedau et al. [5] and fixed building model collections based on line segments by Lee et al. [6]. Our previous work [4] proposed the pop-up 3D plane model, combining CNNs with geometry modeling. Results show that our work is more robust to various corridor configurations and lighting conditions than existing methods.

### B. Multiple Images

*1) v-SLAM using points:* Structure from Motion and v-SLAM have been widely used to obtain 3D reconstructions from images [7]. These methods track image features across multiple frames and build a globally consistent 3D map using optimization. Two representatives of them are direct LSD SLAM [1] and feature- based ORB SLAM [2]. But these methods work poorly in low-texture environments because of the sparse visual and geometric features.

*2) v-SLAM using planes:* Planes or superpixels have been used in [8]–[10] to provide dense mapping in low-texture areas. But they assume camera poses are provided from other sources such as point based SLAM, which may not work well in textureless environments as mentioned above. Recently, Concha et al. [11] also propose to use room layout information to generate depth priors for dense mapping however they don't track and update the room layout thus can only work in small workspace.

*3) Scene understanding:* Some works focus on the scene understanding using multiple images, especially in a Manhattan world. Flint et al. [12] formulate it as Bayesian framework using monocular and 3D features. [13], [14] generate many candidate 3D model hypotheses and subsequently update their probability by feature tracking and point cloud matching. Unfortunately, these methods do not use a plane world to constrain the state estimation and thus cannot solve the problem of v-SLAM in low-texture environments.

## III. SINGLE IMAGE PLANE POP-UP

This section extends our previous work [4] to create a pop-up 3D plane model from a single image. We first briefly recap the previous work, discuss its limitation, and propose two improvements accordingly.

### A. Pop-up 3D Model

There are three main steps in [4] to generate 3D world: CNN ground segmentation (optionally with Conditional Random Field refinement), polyline fitting, and pop up 3D plane model. It outperforms existing methods in various dataset evaluation. However, there are some limitations:

Firstly, [4] fit polylines along the detected ground region which might not be the true wall-ground edges and thus generate a invalid 3D scene model. For example in Figure 2, it cannot model the right turning hallway. This results in problems attempting to use these planes in SLAM framework because even in adjacent frames, the fitted line segments may be different. However, SLAM requires the landmark (in our case, planes), to be invariant across frames.

Secondly, [4] uses a zero rotation pose assumption, which in most cases, is not satisfied. Different rotation angles may generate different pop-up 3D model.

In the following two sections, we solve these problems and generate a more accurate 3D map shown at the bottom of Figure 2.
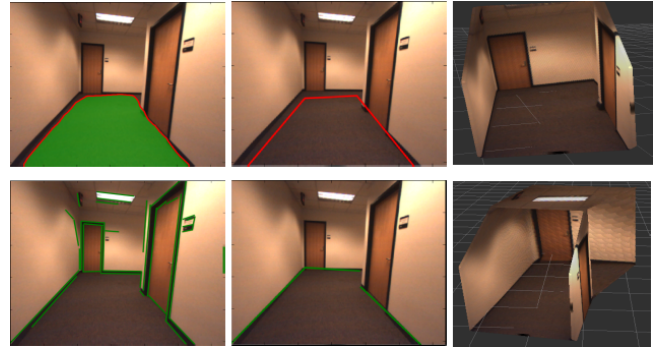


Fig. 2. Single image pop-up plane model. (top) Original method of [4]. From left to right: CNN segmentation, polyline fitting, pop-up 3D model. (bottom) Improved method. From left to right: line segment detection, selected ground-wall edges, pop-up 3D model. Better ground edge detection and camera pose estimation lead to a more accurate 3D model.

### B. Optimal Boundary Detection

Instead of using a fitted polygon as a wall-ground boundary, we propose to detect the true ground-wall edges. We first extract all the line segments using [15]. But as other line detectors, this algorithm also has detection noise. For example, a long straight line may be detected as two disconnected segments. We propose an algorithm to optimally select and merge edges as a wall-ground boundary shown in the bottom center of Figure 2.

Mathematically, given a set of detected edges $V = \{e_1, e_2, ..., e_n\}$, we want to find the optimal subset edges $S \subseteq V$, such that:

$$\max_{S \subseteq V} F(S), \ st \colon S \in I \tag{1}$$

where $F$ is the score function and $I$ is the constraint. Due to the complicated scene structures in the real world, there is no standard way of expressing $F$ and $I$ as far as we know, so we intuitively design them to make it more adaptable to

various environments, not limited to a Manhattan world as it is typically done in many current approaches [11] [13].

The first constraint indicates that edges should be close to the CNN detected boundary curve $\xi$ within a threshold shown as red curve in the top left of Figure 2. It can be denoted as:

$$I_{close} = \{S \colon \forall e \in S, \ dist(e, \xi) < \delta_{close}\} \qquad (2)$$

The second constraint is that edges should not overlap with each other beyond a threshold in image horizontal direction shown in Figure 3(a). This is true for most cases in the real world. In the latter experiments, we find that even for the unsatisfactory configurations in Figure 3(b), our algorithm can select most of the ground edges. We can denote this constraint as:

$$I_{ovlp} = \{S \colon \forall e_i, e_j \in S, O(e_i, e_j) < \delta_{ovlp}\} \qquad (3)$$

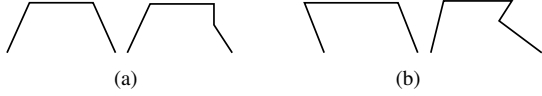where $O$ is horizontal overlapping length between two edges.



Fig. 3. (a) Desired corridor configurations where our algorithm can select all the ground edges. (b) Unsatisfactory configurations because of too much overlap horizontally. Our algorithm might miss some ground edges.

Similarly, we want to maximize the covering of edges in image $x$ direction. So the score function is defined as:

$$F \colon \{S \to \mathbb{R}, \ F = C(S)\} \qquad (4)$$

where $C$ is the horizontal covering length of edge sets $S$.

With the defined score function $F$ and constraints $I = I_{close} \cap I_{ovlp}$, the problem changes to a submodular set optimization. We adopt a greedy algorithm [16] to select the edges in sequence. We initially start with an empty set of edges $S$, then iteratively add edges by:

$$S \leftarrow S \cup \{\underset{e \notin S : S \cup \{e\} \in I}{\arg\max} \ \triangle(e \mid S)\} \qquad (5)$$

until there is no feasible edges. $\triangle(e \mid S)$ is the marginal gain of adding edge $e$ into set $S$. Details and proof of submodularity and optimality are in the appendix.

After getting the edge set $S$, some post processing steps are required for example removing tiny edges and merging adjacent edges into a longer one similar to [5].

### C. Pop-up World from an Arbitrary Pose

**Notations.** We use subscript $w$ to represent global *world* frame and $c$ to denote local *camera* frame. $gnd$ is short for ground plane. A plane can be represented as a homogeneous vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)^\top = (\mathbf{n}^\top, d)^\top$, where $\mathbf{n}$ is the plane normal vector, and $d$ is its distance to the origin [17] [18]. The camera pose is represented by the 3D Euclidean transformation matrix $\mathrm{T}_{w,c} \in \mathbf{SE}(3)$ from local to global frame. Then a local point $\mathbf{p}_c$ can be transformed to global frame by: $\mathbf{p}_w = T_{w,c}\mathbf{p}_c$, and a local plane $\boldsymbol{\pi}_c$ is transformed to global frame by:

$$\boldsymbol{\pi}_w = \mathrm{T}_{w,c}^{-\top}\boldsymbol{\pi}_c \qquad (6)$$

*1) Create 3D model:* For each image pixel $\mathbf{u} \in \mathbb{R}^3$ (homogeneous form) belonging to a certain local plane $\boldsymbol{\pi}_c$, the corresponding 3D pop-up point $\mathbf{p}_c$ is the intersection of backprojected ray $\mathrm{K}^{-1}\mathbf{u}$ with plane $\boldsymbol{\pi}_c$:

$$\mathbf{p}_c = \frac{-d_c}{\mathbf{n}_c^\top(\mathrm{K}^{-1}\mathbf{u})}\mathrm{K}^{-1}\mathbf{u} \qquad (7)$$

where $\mathbf{K}$ is calibration matrix.

Then we show how to compute the plane equation $\boldsymbol{\pi}_c$. Our world frame is built on the ground plane represented by $\boldsymbol{\pi}_{gnd,w} = (0, 0, 1, 0)^\top$. Suppose a ground edge's boundary pixels are $\mathbf{u}_0, \mathbf{u}_1$, their 3D point $\mathbf{p}_{c0}, \mathbf{p}_{c1}$ can be computed by Equation (6) (7). Using the assumption that wall is vertical to the ground, we can compute the wall plane normal by:

$$\mathbf{n}_{wall,c} = \mathbf{n}_{gnd,c} \times (\mathbf{p}_{c1} - \mathbf{p}_{c0}) \qquad (8)$$

We can further compute $d_{wall,c}$ using the constraints that two points $\mathbf{p}_{c0}, \mathbf{p}_{c1}$ lying on the wall.

*2) Camera pose estimation:* The camera pose $\mathrm{T}_{w,c}$ could be provided from other sensors or state estimation methods. Here, we show a single image attitude estimation method which could be used at the SLAM initialization stage. For a Manhattan environment, there are three orthogonal dominant directions $\mathbf{e}_1 = (1, 0, 0)^\top, \mathbf{e}_2 = (0, 1, 0)^\top, \mathbf{e}_3 = (0, 0, 1)^\top$ corresponding to three vanishing points $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ in homogeneous coordinate. If the camera rotation matrix is $\mathbf{R}_{w,c} \in \mathbb{R}^{3\times3}$, then $\mathbf{v}_i$ can be computed by [5] [19]:

$$\mathbf{v}_i = \mathbf{K}\mathbf{R}_{w,c}\mathbf{e}_i, \quad i \in \{1, 2, 3\} \qquad (9)$$

With three constraints of Equation (9), we can recover the 3 DoF rotation $\mathbf{R}_{w,c}$.

### IV. POP-UP PLANE SLAM

This section introduces the *Pop-up Plane SLAM* using monocular images. Plane SLAM has recently been addressed by Kaess [18] with a RGB-D sensor, here we extend it to the monocular case based on the pop-up plane model.

### A. Planar SLAM Formulation

The factor graph of planar SLAM is shown in Figure 4. We need to estimate the 6 DoF camera poses $x_0, ..., x_t$ and plane landmarks $\boldsymbol{\pi}_0, ..., \boldsymbol{\pi}_n$ using the plane measurements $c_0, ..., c_m$, odometry measurements $u_1, ..., u_t$ and initial pose constraint $p$. Note that, our plane landmark also has a label being either ground or wall. The ground plane landmark $\boldsymbol{\pi}_0$ is connected to all pose nodes.

The homogeneous plane representation $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$ is over-parametrized and therefore the information matrix of SLAM is singular and not suitable for Gauss-Newton solver and incremental solvers such as iSAM [20]. We utilize the minimal plane representation in [18] to represent planes as a unit quaternion $\mathbf{q} = (q_1, q_2, q_3, q_4)^\top \in \mathbb{R}^4$ st. $\|q\| = 1$. We can therefore use Lie algebra and exponential map to do plane updates during optimization.
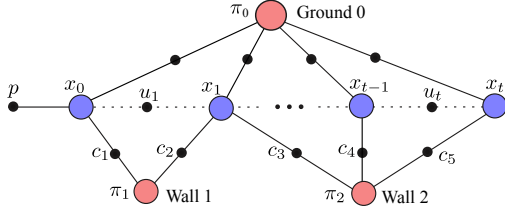
Fig. 4. Plane SLAM factor graph. Variable nodes include camera pose $x$, plane landmark $\pi$. Factor nodes are odometry measurements $u$ and plane measurements $c$. The latter come from the single image pop-up model. Each plane node $\pi$ also has a label of either ground or wall.

### B. Plane Measurement

Most plane SLAM [18], [21] uses RGB-D sensor to get plane measurements $c$ from the point cloud segmentation. In our system, plane measurements $c$ come from the pop-up plane model in Section III-C. Note that the pop-up process depends on the camera pose, more specifically rotation and height because camera $x, y$ position does not affect local plane measurements. So we need to re-pop up the 3D plane model and update plane measurement $c$ after camera poses are optimized by plane SLAM. This step is fast with simple matrix operation explained in Section III-C. It takes less than 1ms to update hundred's plane measurement.

### C. Data Association

We use the following three geometry information for plane matching: the difference between plane normals, plane distance to each other and projection overlapping between planes. The plane's bounding polygon for projection comes from the pop-up process. Outlier matches are first removed by thresholds of the three metrics. Then the best match is selected based on a weighted sum of them.
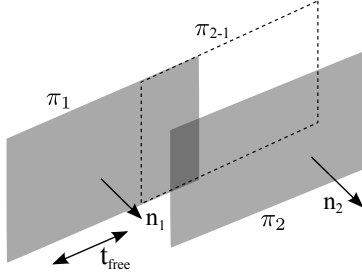


Fig. 5. Data association and unconstrained situations. $\pi_1, \pi_2$ are two planes with normals $n_1, n_2$ respectively. $\pi_{2\_1}$ is the projected plane from $\pi_2$ onto $\pi_1$, used for data association. In this example, $n_1$ and $n_2$ are parallel so there is an unconstrained direction along $t_{free}$.

### D. Loop Closure

We adopt a bag of words (BoW) place recognition method [22] for loop detection. Each frame is represented as a vector of visual worlds computed by ORB descriptors so as to calculate the similarity score between two frames. Once a loop closure frame is detected, we search all the plane pairs in the two frames and find the plane pairs with smallest image space distance. We also tested to keep BoW visual

words for each plane, but it is not robust especially in texture-less images. Different from point landmarks, plane landmarks have different appearance and size in different views. So we may recognize the same planes after the landmark has been created and observed for sometime. So after detecting, for example, $\pi_n$ and $\pi_2$ as the same plane in Figure 6, we shift all the factors of plane $\pi_n$ to the other plane $\pi_2$, and remove the landmark $\pi_n$ from factor graph.
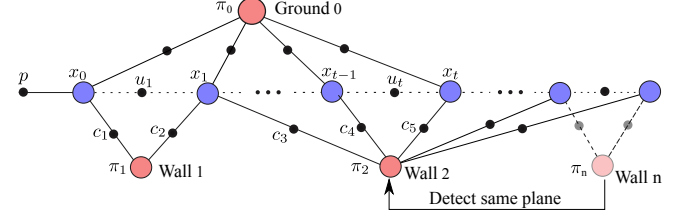


Fig. 6. Plane SLAM loop closure. After detecting a loop closure, we shift all the factors of plane $\pi_n$ to $\pi_2$ and remove $\pi_n$ from factor graph.

## V. POINT-PLANE SLAM FUSION

Compared to point based SLAM, planar SLAM usually contains much fewer landmarks so it becomes easily unconstrained. For example in a long corridor in Figure 5 where left and right walls are parallel, there is a free unconstrained direction $t_{free}$ along corridor if there is no other plane constraints. We solve this problem by incorporating with point based SLAM, specifically LSD SLAM [1], to provide photometric odometry constraints along the free direction. We propose the two following combinations.

### A. Depth Enhanced LSD SLAM

This section shows that scene layout understanding could boost the performance of traditional SLAM. LSD SLAM has three main threads: camera tracking, depth estimation and global optimization in Figure 7. The core part is depth estimation, determining the quality of other modules. In LSD SLAM, when a new depth map of a keyframe is created, it propagates some pixels' depth from the previous keyframe if it is available. Then the depth map is continuously updated by new frames using multiple-view stereo (MVS). Since our single image pop-up model in Section III provides each pixel's depth estimation, we integrate its depth into LSD depth map in the following way:

(1) If a pixel has no propagated depth or the variance of the LSD SLAM depth exceeds a threshold, we directly use pop-up model depth.

(2) Otherwise, if a pixel has a propagated depth $d_l$ with variance $\sigma_l^2$ from LSD, we fuse it with the pop-up depth $d_p$ of variance $\sigma_p^2$ using the filtering approach [23]:

$$\mathcal{N} \left( \frac{\sigma_l^2 d_p + \sigma_p^2 d_l}{\sigma_l^2 + \sigma_p^2}, \frac{\sigma_l^2 \sigma_p^2}{\sigma_l^2 + \sigma_p^2} \right) \qquad (10)$$

$\sigma_p^2$ could be computed by error propagation rule during the pop-up process. In Section III-C, the pixel uncertainty of $\mathbf{u}$ can be modeled as bi-dimensional standard Gaussians $\Sigma_u$. If

the Jacobian of $\mathbf{p}_c$ *wrt.* $\mathbf{u}$ is $J_u$ from Equation (7), then the 3D point's covariance $\Sigma_{p_c} = J_u \Sigma_u J_u^\top$. We find that depth uncertainty $\sigma_p^2$ is proportional to the depth square, namely $\sigma_p^2 \propto d_p^2$.
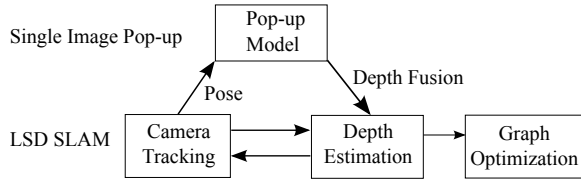


Fig. 7. *Depth Enhanced LSD SLAM* algorithm that integrates depth estimates from the pop-up model.

Depth fusion could greatly increase the depth estimation quality of LSD SLAM especially at the initial frame where LSD SLAM just randomly initializes the depth and at the low parallax scenes where MVS depth triangulation has low quality. This is also demonstrated in the latter experiments.

### B. LSD Pop-up SLAM

There has been some work jointly using point and plane as landmarks in one SLAM framework [21] using RGB-D sensors. Currently, we propose a simple version of it to run two stages of SLAM methods. The first stage is *Depth Enhanced LSD SLAM* in Section V-A. We then use its pose output as odometry constraints to run a plane SLAM in Section IV. The frame-to-frame odometry tracking based on photometric error minimization could provide constraints along the unconstrained direction in plane SLAM and can also capture the detailed fine movements, demonstrated in the latter experiments.

Figure 8 shows the relationship of the three SLAM methods in this paper. The blue dashed box is the improved LSD SLAM: *Depth Enhanced LSD SLAM*. The green and red box show two kinds of plane SLAM. The difference is that *LSD Pop-up SLAM* in this section has additional odometry measurement while *Pop-up Plane SLAM* does not have and usually uses a constant velocity assumption.
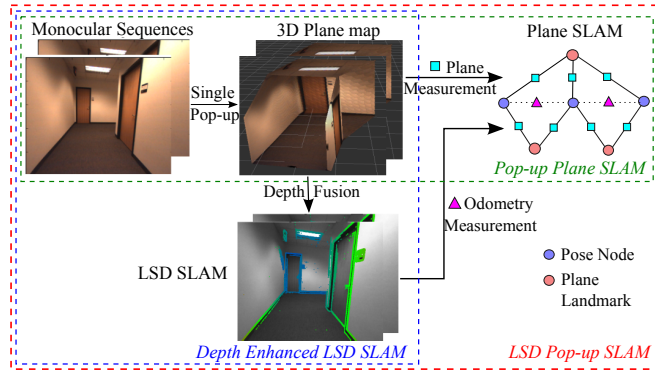


Fig. 8. Relationship between three proposed SLAM methods: (1) *Pop-up Plane SLAM* uses plane measurements from the single image pop-up model. (2) *Depth Enhanced LSD SLAM* is LSD SLAM with depth fusion from the pop-up model. (3) *LSD Pop-up SLAM* is plane SLAM with additional odometry measurements from *Depth Enhanced LSD SLAM*.

## VI. EXPERIMENTS AND RESULTS

We test our SLAM approaches on both the public TUM dataset [24] and two collected corridor datasets to evaluate the accuracy and computational cost. Results can also be found in the supplemental videos. We compare the state estimation and 3D reconstruction quality with two state-of-art point based monocular SLAM approaches: LSD SLAM [1] and ORB SLAM [2].

### A. TUM SLAM dataset

We choose the TUM *fr3/structure_notexture_far* dataset in Figure 1, which is a challenging environment composed of five white walls and a ground plane. We only use RGB images for experiments and use the depth images for evaluation.

*1) Qualitative Comparison:* Unfortunately, neither LSD nor ORB SLAM work in this environments because there are only few features and high gradient pixels.

For the *Pop-up Plane SLAM* in Section IV, we use the ground truth pose for initialization and a constant velocity motion assumption as odometry measurements. Since the initial truth height is provided, the pop-up model has an absolute scale. Therefore, we can directly compare the pose and map estimates with ground truth without any scaling. The constructed 3D map is shown in Figure 1.

*2) Quantitative Comparison:* The absolute trajectory estimate is shown in Figure 9. This dataset has a total length of 4.58m and our mean positioning error is $0.18 \pm 0.07$m, with endpoint error $0.10$m. From Figure 9, our algorithm captures the overall movement but not the small jerk movement in the middle. This is mainly due to the fact that there are only few plane landmarks in SLAM. In addition, *Pop-up Plane SLAM* does not have frame-to-frame odometry tracking to capture the detailed movement, which is commonly used in point based SLAM. In the latter experiments, we show that after getting odometry measurements, state estimation of *LSD Pop-up SLAM* improves greatly.
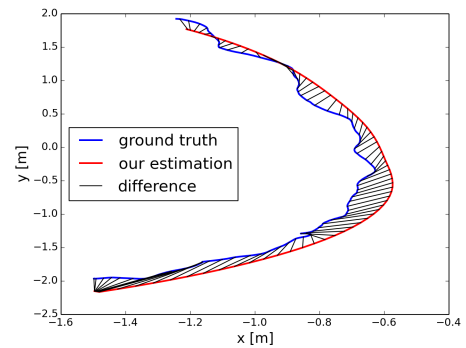


Fig. 9. Absolute trajectory estimation using *Pop-up Plane SLAM* on TUM *fr3/str_notex_far* dataset. The positioning error is 3.9%, while LSD SLAM and ORB SLAM both fail.

To evaluate mapping quality, we use provided depth maps to compute the ground truth plane position by point cloud plane segmentation using the PCL RANSAC algorithm. The plane normal error is only $2.8°$ as shown in Table I. We then re-project the 3D plane model onto images to get each pixel's

depth estimates. The evaluation result is shown in Figure 10 and Table I. The mean pixel depth error is 6.2 cm and 86.8% of the pixels' depth error is within 0.1m.
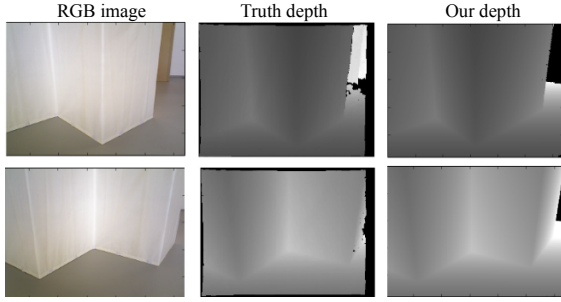


Fig. 10. Depth reconstruction comparison on TUM dataset.

TABLE I
3D RECONSTRUCTION EVALUATION ON TUM DATASET.

|  | Plane normal error | Depth error | Depth error < 0.1m |
|---|---|---|---|
| Value | 2.83° | 6.2 cm | 86.8% |

### B. Large Indoor Environment

In this section, we present experimental results using a hand-held monocular camera with a resolution of $640 \times 480$ in two large low-texture corridor environments. The camera has a large field of view ($\sim 90°$) which LSD and ORB SLAM typically prefer. Since we do not have ground truth depth or pose, we only evaluate the loop closure error and qualitative map reconstruction. The pose initialization uses the single image rotation estimation in Section III-C with an assumed height of 1m.

*1) Corridor dataset I:* The first dataset is shown in Figure 11. LSD SLAM, top center, does not perform well. The best result for ORB SLAM is shown in the top right. Through the tests, we find that even using the same set of parameters, ORB SLAM often cannot initialize the map and fails to track cameras. The randomness results from the RANSAC map initialization of ORB SLAM [2].

Since actual long corridors are easily under-constrained, *Pop-up Plane SLAM* with no actual odometry measurement does not work well. We only provide results for the two other SLAM methods introduced in Section V. *Depth Enhanced LSD SLAM* generates a much better map as shown in the bottom left of Figure 11 compared to the original LSD SLAM. Though it is a semi-dense map, we can clearly see the passageway and turning. Based on that, the *LSD Pop-up SLAM* generates a dense 3D model with distinct doors and pillars.

*2) Corridor dataset II:* The second dataset is a 60m square corridor containing a large loop shown in Figure 12. ORB SLAM generates a better map than LSD SLAM, but it does not start tracking until it comes to a large open space with more features and enough parallax.
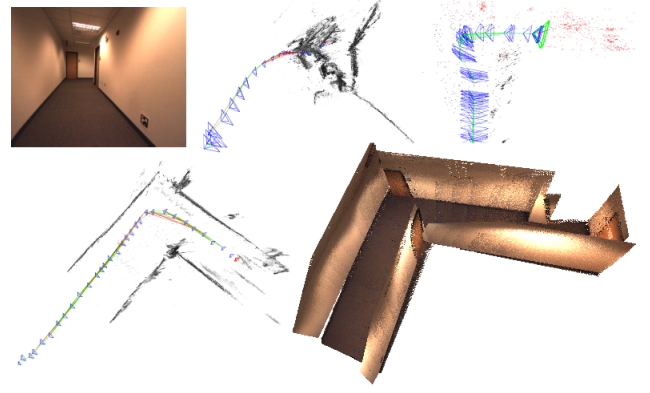


Fig. 11. Corridor dataset I. (top) From left to right: sample frame, LSD SLAM result, ORB SLAM result. (bottom) Our *Enhanced LSD SLAM* in Section V-A, *LSD Pop-up SLAM* result in Section V-B.

The result of our algorithms is shown in Figure 13 where the red line is *Enhanced LSD SLAM* and green line is *LSD Pop-up SLAM*. With the automatic loop closure detection, the *LSD Pop-up SLAM* generates the best 3D map and smallest loop closure error. The grid dimension is $1 \times 1m^2$ in Figure 13 and the loop closure positioning error is 0.4m of the total 60m trajectory.
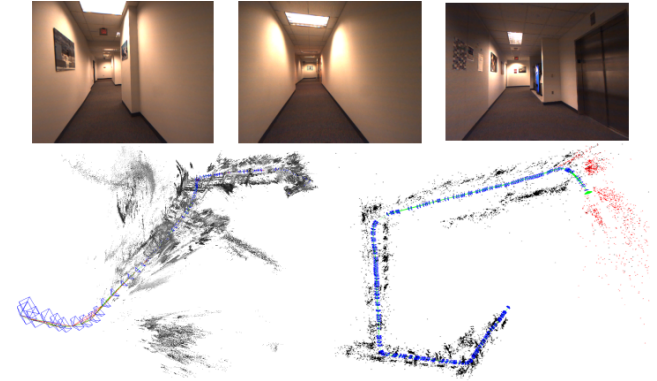


Fig. 12. Corridor dataset II with loop closure. (top) Sample frames in the dataset. (bottom): LSD SLAM result, ORB SLAM result.

### C. Runtime Analysis

Finally, we provide the computation analysis of the Corridor dataset II in Table II. All timings are measured on CPU (Intel i7, 4.0 GHz) and GPU (only for CNN). Most of the code is implemented in C++. Currently, the CNN segmentation, edge detection, and selection consumes 30ms. Note that compared to CNN model in [4], we change the fully connected layers from 4096 to 2048 to reduce the segmentation time by half without affecting the accuracy too much. The iSAM incremental update takes 17.43ms while batch optimization takes 45.35ms. Therefore we only use batch optimization to re-factor the information matrix when a loop closure is detected. In all, our plane SLAM algorithm can run in real time at over 20Hz using single-threaded programming.
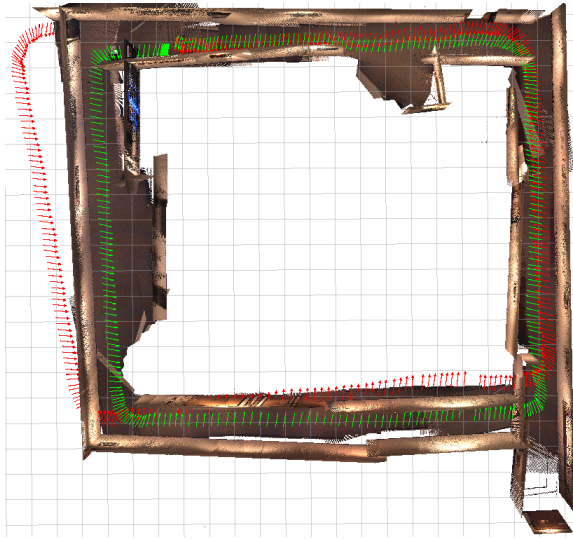
Fig. 13. Corridor dataset II with loop closure (continued). Red line: *Enhanced LSD SLAM* result in Section V-A, green line: *LSD Pop-up SLAM* result in Section V-B. Loop closure happens around the top left corner. The grid dimension is 1m. Loop closure positioning error is 0.67%.

We also note that unlike point landmarks, a plane landmark can be observed in many adjacent frames, so we actually do not need to pop up planes for each frame. Thus in all the above pop-up experiments, we process the images at 3Hz (every 10 images), which we find is enough to capture all the planes. This is similar to the keyframe techniques used in many point-based SLAM algorithms.

TABLE II
SLAM STATISTICS AND TIME ANALYSIS ON CORRIDOR DATASET II.

| | |
|---|---|
| Number of planes | 146 |
| Number of poses | 344 |
| Number of factors | 1974 |
| CNN segmentation (ms) | 17.8 |
| Edge detection and selection (ms) | 13.2 |
| Data association (ms) | <1 |
| iSAM optimization (ms) | 17.4 |
| Total frame time (ms) | **49.4** |

### D. Discussion

*1) Height effect on map scale:* Unlike RGB-D plane SLAM whose plane measurements are generated by the actual depth sensor, our plane measurement and scale is determined by the camera height in the pop-up process in Section III-C. Camera height cannot be constrained by the plane measurements any more and therefore can only be constrained by other information such as odometry measurements or other sensors such as IMU. If other information is inaccurate or inaccessible, the map scale and camera height

might drift using the plane SLAM alone. During all the experiments, we did not encounter the scale drift problem because the cameras are kept at a nearly constant height. In the future, we would like to integrate with other sensors.

*2) Ground effect on graph complexity:* Since the pop-up process in Section III-C requires the ground plane to be visible, the ground plane landmark is connected to all the camera poses as shown in Figure 4. This will reduce the sparsity of the information matrix and increase the computational complexity in theory. However, it can be alleviated by *variable reordering* before matrix decomposition, for example using the COLAMD algorithm that will force this variable towards the last block column, thereby reducing its impact (see [20]). From the experiments, the ground plane usually increases *fill-in* (added non-zero entries) by only about 10%.

## VII. CONCLUSIONS

In this paper, we propose *Pop-up Plane SLAM*, a real-time monocular plane SLAM system combined with scene layout understanding. It is especially suitable for low-texture environments because it can generate a rough 3D model even from a single image. We first extend previous work to pop up a 3D plane world from a single image by detecting the ground-wall edges and estimating camera rotation. Then, we formulate a plane SLAM approach to build a consistent plane map across multiple frames and also provide good state estimates. The plane landmark measurement comes from the single image pop-up model. We utilize the minimal plane representation for optimization and also implement plane SLAM loop closing.

Since plane SLAM itself is easily under-constrained in some environments, we propose to combine it with point based LSD SLAM in two ways: the first is *Depth Enhanced LSD SLAM* by integrating pop-up pixel depth into LSD depth estimation, the second is *LSD Pop-up SLAM*, which uses poses from *Depth Enhanced LSD SLAM* as odometry constraints and runs a separate *Pop-up Plane SLAM*.

In the experiment with the public TUM dataset, *Pop-up Plane SLAM* generates a dense 3D map with depth error of 6.2 cm and state estimates error of 3.9% while the state-of-art LSD or ORB SLAM both fail. Two collected large corridor datasets are also used to demonstrate its practicality and advantages. The loop closure error of *LSD Pop-up SLAM* on a 60m long dataset is only 0.67%, greatly outperforming LSD and ORB SLAM methods. The runtime analysis demonstrates that our algorithm could run in (near) real-time over 10Hz.

In the future, we want to combine point, edge and plane landmarks in a unified SLAM framework. In addition, we would like to test this algorithm on robots. Besides, more work needs to be done in clutter corridors where ground-wall boundaries may be occluded.

APPENDIX

*A. Submodular Edge Selection*

Here, we briefly provide the optimality analysis of the edge selection as an extension to Section III-B. We prove that it is a submodular set selection problem with matroid constraints [16].

*1) Monotonicity:* The score function $F$ in Equation (4) is obviously monotonically increasing because adding more edges, the covering in image horizontal direction will not decrease.

*2) Submodularity:* We first define the marginal gain of $e$ *wrt.* $S$ as the increase of score $F$ after adding element $e$ into $S$, namely

$$\triangle(e \mid S) := F(S \cup \{e\}) - F(S)$$

For two sets $S_1 \subset S_2$, edge $e$ may overlap with more edges in $S_2$ and thus reduce the marginal gain compared to $S_1$, so it satisfies the submodularity condition:

$$\triangle(e \mid S_1) \geq \triangle(e \mid S_2), \ \forall S_1 \subseteq S_2$$

*3) Matroid constraint type:* We can remove the edges that are far from CNN boundary before submodular optimization, so we only consider the second constraint $I_{ovlp}$ in Equation (3). Denote all the conflicting edge pairs as $E_i = \{(e_{i1}, e_{i2}) \mid O(e_{i1}, e_{i2}) \geq \delta_{ovlp}\}, i = 1, 2, ..., k$. For each $E_i$, we form a partition of the ground set $V$ by two disjoint sets $P_i = \{E_i, V \setminus E_i\}$ and thus can form a partition matroid constraint $I_i^m = \{S : |S \cap P_i^1| \leq 1, |S \cap P_i^2| \leq n\}$, where $P_i^1$ and $P_i^2$ are two elements of $P_i$. This is because we can pick at most one element from $E_i$. The union of $k$ such separate matroid constraints forms the original constraint $I_{ovlp} = I_1^m \cap I_2^m ... \cap I_k^m$.

*4) Optimality:* From [16], the greedy algorithm in Equation (5) of the submodular optimization with matroid constraints is guaranteed to produce a solution $S$ such that $F(S) \geq \frac{1}{k+1} \max_{S \subseteq I} F(S)$. It is also important to note that this is only a worst case bound and in most cases, the quality of solution obtained will be much better than this lower bound.

ACKNOWLEDGMENTS

REFERENCES

[1] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular slam. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[2] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *Robotics, IEEE Transactions on*, 31(5):1147–1163, 2015.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[4] Shichao Yang, Daniel Maturana, and Sebastian Scherer. Real-time 3D scene layout from a single image using convolutional neural networks. In *Robotics and automation (ICRA), IEEE international conference on*, pages 2183 – 2189. IEEE, 2016.

[5] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1849–1856. IEEE, 2009.

[6] Daniel C Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2136–2143. IEEE, 2009.

[7] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality (ISMAR), 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010.

[9] Alejo Concha and Javier Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 5686–5693. IEEE, 2015.

[10] Pedro Pinies, Lina Maria Paz, and Paul Newman. Dense mono reconstruction: Living with the pain of the plain plane. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 5226–5231. IEEE, 2015.

[11] Alejo Concha, Wajahat Hussain, Luis Montano, and Javier Civera. Incorporating scene priors to dense monocular mapping. *Autonomous Robots*, 39(3):279–292, 2015.

[12] Alex Flint, David Murray, and Ian Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *Computer Vision (ICCV), IEEE International Conference on*, pages 2228–2235. IEEE, 2011.

[13] Grace Tsai, Changhai Xu, Jingen Liu, and Benjamin Kuipers. Real-time indoor scene understanding using bayesian filtering with motion cues. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 121–128. IEEE, 2011.

[14] Axel Furlan, Stephen Miller, Domenico G Sorrenti, Li Fei-Fei, and Silvio Savarese. Free your camera: 3d indoor scene understanding from arbitrary camera motion. In *British Machine Vision Conference (BMVC)*, page 9, 2013.

[15] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):722–732, 2008.

[16] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

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

[19] Carsten Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9):647–655, 2002.

[20] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, 2008.

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

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

[23] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456, 2013.

[24] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D slam systems. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

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