2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Keyframe-based Dense Planar SLAM

Ming Hsiao*†, Eric Westman*†, Guofeng Zhang‡ and Michael Kaess†

***Abstract*— In this work, we develop a novel keyframe-based dense planar SLAM (KDP-SLAM) system, based on CPU only, to reconstruct large indoor environments in real-time using a hand-held RGB-D sensor. Our keyframe-based approach applies a fast dense method to estimate odometry, fuses depth measurements from small baseline images, extracts planes from the fused depth map, and optimizes the poses of the keyframes and landmark planes in a global factor graph using incremental smoothing and mapping (iSAM). Using the fast odometry estimation, correct plane correspondences may be found projectively, and the pose of each frame can be estimated accurately even without sufficient planes to fully constrain the 6 degree-of-freedom transformation. The depth map generated from the local fusion process generates higher quality reconstructions and plane segmentations by eliminating noise. Moreover, explicitly modeling plane landmarks in the fully probabilistic global optimization significantly reduces the drift that plagues other dense SLAM algorithms. We test our system on standard RGB-D benchmarks as well as additional indoor environments, demonstrating its state-of-the-art performance as a real-time dense 3D SLAM algorithm, *without* the use of GPU.**

I. Introduction

The capabilities of visual SLAM algorithms have greatly expanded in recent years. Sparse or semi-dense 3D reconstructions and accurate trajectories may be generated over large-scale scenes using monocular cameras—although they can only be accurate up to a scale factor [20, 7]. Dense 3D reconstructions may also be generated using RGB-D sensors by fusing data from every frame into the model, a process that must be accelerated by a GPU in order to achieve real-time performance [21, 15, 28, 29]. Both of these approaches suffer from drift accumulation, which may be partially compensated for by introducing explicit loop closure constraints, which is a common practice in practically all modern SLAM systems. However, when mapping indoor scenes or other highly planar environments, planes may be directly modeled as landmarks in order to further constrain the camera motion and significantly reduce drift. Although a planar SLAM solution might not be able to preserve the details of non-planar regions very well compared to unstructured dense SLAM methods, [5, 17, 28, 29], the planar regions can still preserve the general geometric structure of indoor environments in a much cheaper dense representation.

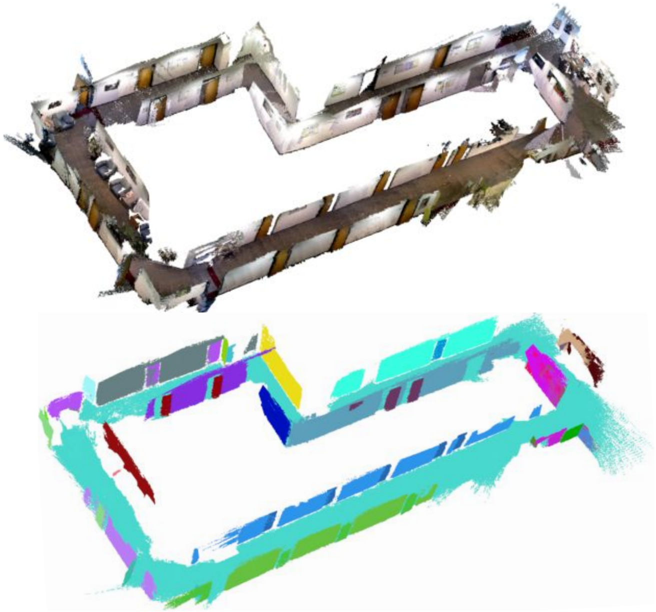


Fig. 1: Our KDP-SLAM system can reconstruct large indoor environments with loops. Top: An example of dense pointcloud map. Bottom: Dense map with false-colored planes.

While various algorithms have been proposed recently for performing planar SLAM with monocular or RGB-D data (to be discussed in detail in section II), there remain key shortcomings which limit their applicability. All known dense planar SLAM algorithms in the literature require GPU acceleration for real-time (30 fps) performance, which generally prohibits the use of such systems on mobile devices or robots. In real-world environments, there are often insufficient planar features to fully constrain the camera motion, which can cause large amounts of drift. Lastly, recent planar SLAM work has demonstrated reconstructions only on a small scale (room-size rather than building-size). In this paper, we propose a keyframe-based dense planar SLAM system that addresses all of these problems in a computationally efficient manner.

First of all, we apply a keyframe-based framework in our planar SLAM solution to achieve real-time operation on a CPU. This structure eliminates the need to extract planes and perform factor graph optimization at every frame, which are the most expensive components of this system.

Obtaining accurate pose estimates at every frame is essential to our mapping algorithm. Feature point-based odometry methods such as Fovis [12] perform poorly in scenes absent

*Joint first authors

†Ming Hsiao, Eric Westman, and Michael Kaess are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {mhsiao, westman, kaess}@cmu.edu

‡Guofeng Zhang is with the State Key Lab of CAD & CG, Zhejiang University, Hangzhou, Zhejiang 310058, China. zhangguofeng@cad.zju.edu.cn

This work was partially supported by the Office of Naval Research under awards N00014-16-1-2365 and N00014-16-1-2103, and by the National Science Foundation under award IIS-1426703.

978-1-5090-4633-1/17/$31.00 ©2017 IEEE

of rich texture, which are commonly encountered in indoor environments with large planar structures. On the other hand, dense methods, such as iterative closest point (ICP) [4] and dense RGB-D odometry [24, 16], can be more robust given that they take the entire image and 3D structure into account. However, because of the expensive calculations over all the points and pixels, dense methods usually require parallel computing using GPU for real-time applications [28, 29]. To achieve higher efficiency, we develop an odometry algorithm that combines (1) an original geometric alignment method in the vein of ICP that utilizes planar regions and (2) concepts from the semi-dense and dense visual odometry literature [6, 16, 24]. Our algorithm runs faster than real-time on CPU only and is suitable for our keyframe-based planar SLAM approach.

Using the pose estimates from our odometry algorithm, depth information from small baseline images is fused in a local map to reconstruct dense 3D structure and extract accurate plane models. Additionally, we develop a novel plane association algorithm with a point-to-model matching criteria, which considerably improves the accuracy of plane matches. A loop closing method is also integrated into our system, which conducts place recognition first and then closes the loop with both keyframe-to-keyframe and plane-to-plane constraints.

In summary, there are four key contributions in this work:

1) We develop a fast dense RGB-D odometry algorithm based on planar structure and concepts from existing visual odometry methods.
2) We generate a local depth map to improve the quality of the 3D reconstruction and plane extraction.
3) We introduce a novel projective data association algorithm for matching planes.
4) Our system is the first known dense planar SLAM algorithm capable of real-time (30 fps) CPU only execution.

In the following section we discuss related work. In section III we present the multi-threaded structure of our planar SLAM system. In section IV we introduce the fast dense odometry method and its details in implementation. Section V describes the local dense fusion algorithm and explains how it benefits plane extraction. In section VI we highlight the key components for planar SLAM with factor graph, including mathematical representation, optimization, data association and loop closing. Section VII shows experimental results and comparisons. Finally we summarize the contributions and discuss future work in section VIII.

II. Related Work

The idea of using planar features to solve SLAM problems has been studied often in recent years. Some earlier works [22, 27, 18] extract planes from range sensors for mapping tasks. Though these do not utilize hand-held RGB-D sensors, they demonstrate the feasibility of SLAM with planar structures.

In our previous work [13], a quaternion-based minimal plane representation is utilized to update the planes during optimization without encountering singularities. However, it assumes the plane measurements fully constrain the camera motion, which is often not the case in real-world environments. Point-plane SLAM [26] used the combination of three point/plane primitives to fully constrain camera poses and generate a plane-based 3D model. In contrast to only using one or two feature points to constrain the unconstrained directions from insufficient plane associations, we make use of a dense odometry method to fully constrain the poses first and later add in planar constraints among keyframes to further enhance the robustness of the system. Dense planar SLAM [23] uses a dense ICP method to estimate sensor poses which requires GPU for real-time computation. Although it extracts planes from a global dense map generated by point-based fusion [15], the planes are not used as landmarks to optimize the camera poses globally.

CPA-SLAM [19] also requires GPU for tracking towards both keyframes and plane models. Although it applies a soft labeling technique to reduce the effect of incorrect plane segmentation on the pose estimate, the plane model can still be wrong in the global optimization. While our method hard labels segmented planes, the planes extracted from the smoothed depth map are noticeably more accurate than those extracted from the raw data.

A variety of general-purpose dense SLAM algorithms have been presented in the literature which do not explicitly model planes. In this domain, ElasticFusion [29] and Kintinuous [28] are several state-of-the-art GPU-accelerated methods. However, ElasticFusion is designed for integrating many images of the same small-scale scene from different viewpoints, rather than for large-scale mapping. And even though Kintinuous has successfully run on some of the largest environments found in the literature, we can do better by exploiting the structure of indoor environments, as shown in the results. More recently, the Matterport [1] scanner has been developed to generate very accurate 3D reconstructions, but it uses an offline algorithm to align multiple 360° point-clouds and usually takes hours for processing. In contrast, our proposed system is a fully online, CPU-based SLAM system which provides accurate camera trajectories and 3D models of indoor environments.

III. System Structure

Our KDP-SLAM system consists of three concurrent threads (see Fig. 2): (1) the fast dense RGB-D odometry method and frame labeling process, (2) the selective local depth fusion algorithm, and (3) global planar mapping.

The fast dense odometry method estimates the pose of every frame relative to that of the most recent *reference frame* R_j. Precise transformations (solid black lines in Fig. 2-a) are estimated for specially selected frames: *keyframes*, reference frames, and *fusion frames*. Rough transformations (dotted black lines in Fig. 2-a) are estimated for all other frames. If the new frame's estimated pose is too far away from that of R_j, it is set as the next reference frame R_{j+1}.

Each *local interval* L_i, which extends from keyframe K_i through the last frame before keyframe K_{i+1}, maintains

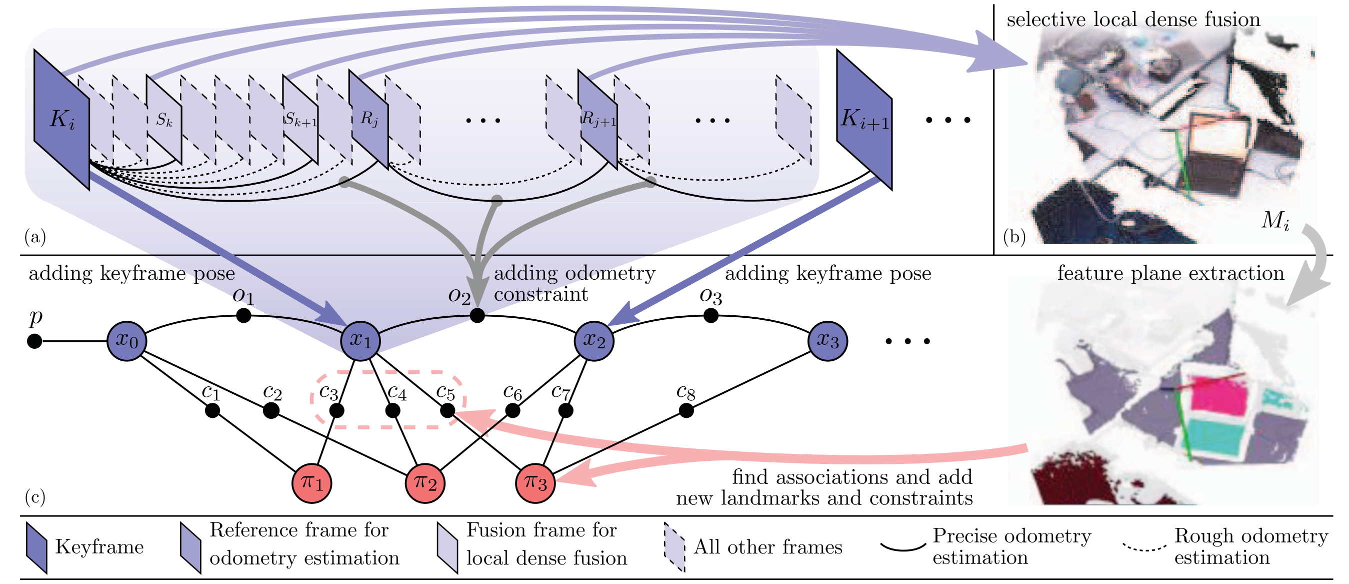


Fig. 2: The KDP-SLAM system consists of three concurrent threads: (a) fast dense RGB-D odometry algorithm and frame labeling process, (b) selective local depth fusion algorithm, and (c) global planar mapping. Note that for the set of all keyframes $\mathcal{K}$, all reference frames $\mathcal{R}$, and all fusion frames $\mathcal{U}$, $\mathcal{K} \subset \mathcal{R} \subset \mathcal{U}$ holds. Also, the loop closure constraint factors between keyframes are not shown in the figure for readability.

a distinct depth map which is initialized with the depth image of K_i. Depth measurements from selected fusion frames within L_i are projected into and fused with the depth map using the frame's precisely estimated pose. A frame is selected as a fusion frame if its pose is too far from that of the previous fusion frame. In principle, every frame could be selected as a fusion frame, but in practice they must be sparsely selected in order to achieve real-time performance.

Selection of a new keyframe K_{i+1} (based on the pose relative to K_i) launches the global planar mapping thread. This processes the previous local map L_i in order to: segment planes, fit planes to point clusters, perform data association, update the factor graph, perform global optimization, and find loop closures. The pose x_i of K_i and the states $\pi_p, ..., \pi_{p'}$ of the relevant landmark planes $\Pi_p, ..., \Pi_{p'}$ are added into a factor graph as variable nodes and linked with each other by factors. The factors between keyframes encode odometry constraints $o_1, ..., o_t$, and those between keyframes and planes encode plane observations $c_1, ..., c_q$ (see Fig. 2-c). By accumulating the relative pose estimations along the references frames in the local interval, we can calculate the relative transformation between consecutive keyframes (the gray arrows from Fig. 2-a to 2-c) and add it into the graph as an odometry constraint. The system applies iSAM [14] to update the graph incrementally whenever a pose node of a new keyframe is added into the factor graph with its corresponding factor nodes. When a loop is detected, the entire graph is batch optimized.

IV. Fast Dense RGB-D Odometry

At every new frame F_n we estimate the camera pose relative to the most recent reference frame R_r. Each frame's pose is initialized using the previous frame's pose. For fusion frames (and therefore also reference frames and keyframes), we estimate poses using a precise method, and for all other frames we estimate rough transformations for the sake of computational efficiency. For both rough and precise odometry estimation we integrate one geometric method and one photometric method into a combined optimization problem as in [28, 29], since these two kinds of methods are complementary and can achieve better accuracy and robustness together:

$$E_\text{total} = E_\text{geo} + \lambda E_\text{pho}. \tag{1}$$

We combine our novel *iterative projected plane (IPP)* method (as the geometric component) and a pyramid dense RGB-D odometry method [24, 16] using Laplacian images (as the photometric component) to estimate the rough odometry. Similarly, we combine IPP with our *semi-dense RGB-D odometry (SRO)* method, as the geometric and photometric components, respectively, to estimate the precise odometry. The weighting λ is used to adjust the relative importance of the two error terms in the optimization. The remainder of this section describes these three methods in detail.

A. Iterative Projected Plane (IPP)

The basic idea of IPP algorithm is to only use the planar regions for 3D registration, which allows a much faster pose estimation if sufficient planar regions are observed in both frames, F_n and R_r. First, planes are fitted to small regions from all over the depth image of each frame. Then, camera projection is applied to find associations between the planes in the two frames. By minimizing the distances between the associated plane pairs, the relative transformation between the two frames can be updated iteratively until convergence. However, it is hard to extract accurate and robust small planar regions efficiently from the noisy raw depth data.

Therefore we preprocess the depth image to reduce noise in the potentially planar regions before extracting the planes.

In general indoor scenes, if a region is smooth in intensity (color) it is likely to be smooth in depth as well. As a result, we iteratively smooth the depth image in areas with low intensity gradient with a three-pixel kernel. Note that we do not smooth the entire depth image so that the non-planar regions with strong geometric features (e.g. edges and corners) will not be averaged out and mistaken as planar regions.

After the depth smoothing, we extract planes from small regions in the partially-smoothed depth image. First, we uniformly divide the depth image into small grids (10×10 pixels in our implementation). In each grid, we uniformly sample several points and calculate the local normal directions of those points. If more than a threshold of normal vectors are close to parallel, we use these points and normal vectors to generate a plane model, find inlier points in the same grid, and refine the plane model using all the inliers (see Fig. 3-a). A plane $\pi_\text{n}^{[i]}$ in F_n is corresponded to a plane $\pi_\text{r}^{[i]}$ in R_r if the projection of the center point $\mathbf{c}_\text{n}^{[i]} = \begin{bmatrix} x_\text{n}^{[i]} & y_\text{n}^{[i]} & z_\text{n}^{[i]} \end{bmatrix}^\top$ of $\pi_\text{n}^{[i]}$ onto R_r is within the grid that contains $\pi_\text{r}^{[i]}$ given the current estimation of the relative transformation between these two frames. With m plane correspondences found in any iteration, the relative transformation between the two frames can be found by minimizing the geometric error:

$$E_\text{geo} = \sum_{i=1}^{m} \left\| \left(R(\boldsymbol{\xi}) \mathbf{c}_\text{n}^{[i]} + \mathbf{t}(\boldsymbol{\xi}) \right)^\top \mathbf{n}_\text{r}^{[i]} + d_\text{r}^{[i]} \right\|^2 \tag{2}$$

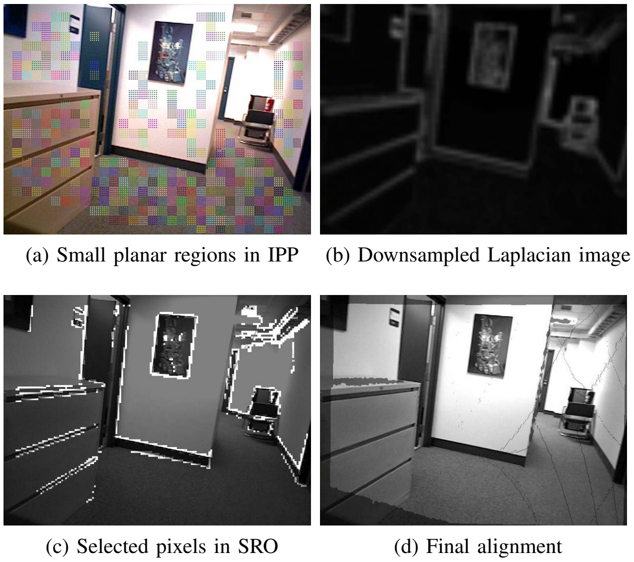
$$\approx \sum_{i=1}^{m} \left\| A^{[i]} \boldsymbol{\xi} - e^{[i]} \right\|^2 , \tag{3}$$

where $A^{[i]} = \left[\mathbf{n}_\text{r}^{[\text{i}]} \right]_\times \mathbf{c}_\text{n}^{[\text{i}]}$ and $e^{[i]} = -\mathbf{n}_\text{r}^{[i]\top} \mathbf{c}_\text{n}^{[i]} - d_\text{r}^{[i]}$. $\boldsymbol{\xi} = \begin{bmatrix} \beta & \gamma & \alpha & t_x & t_y & t_z \end{bmatrix}^\top$ is the minimal representation of the desired 6 DoF relative transformation with three rotations (β, γ, α) and three translations (t_x, t_y, t_z). $R(\boldsymbol{\xi})$ and $\mathbf{t}(\boldsymbol{\xi})$ are the corresponding rotation matrix and translation vector of $\boldsymbol{\xi}$. $\mathbf{c}_\text{n}^{[i]}$ is approximated as the intersection of $\pi_\text{n}^{[i]}$ and the projection ray that goes through the center pixel of the grid that contains $\pi_\text{n}^{[i]}$. $\mathbf{n}_\text{r}^{[i]} = \begin{bmatrix} a_\text{r}^{[i]} & b_\text{r}^{[i]} & c_\text{r}^{[i]} \end{bmatrix}^\top$ is the unit normal vector and $d_\text{r}^{[i]}$ is the distance parameter of the plane $\pi_\text{r}^{[i]}$,

$$a_\text{r}^{[i]} x + b_\text{r}^{[i]} y + c_\text{r}^{[i]} z + d_\text{r}^{[i]} = 0, \tag{4}$$

that corresponds with $\pi_\text{n}^{[i]}$. Note that Eq. 3 is an approximation of Eq. 2 when the transformation is small, which is a valid assumption for most 30 fps hand-held SLAM problems. IPP can run at about 100 fps on a single thread of a CPU due to the projective plane association and the relatively small number of planes compared to the number of points in the raw image.

Since IPP relies on planar surfaces only, it cannot find accurate pose estimations alone if sufficient planes are not observed and matched in the scene (e.g. cluttered scene without enough planar surfaces). As a result, we integrate other photometric methods with IPP to solve this problem.


(a) Small planar regions in IPP (b) Downsampled Laplacian image

(c) Selected pixels in SRO (d) Final alignment

Fig. 3: A rough odometry transformation is estimated by combining IPP (a) and pyramid odometry with Laplace images (b). Using this pose as an initialization point, we can combine IPP (a) and SRO (c) to estimate a more precise transformation (d).

B. Pyramid Dense RGB-D Odometry

Our pyramid dense RGB-D odometry method mostly follows [24, 16] except that we utilize the Laplacian of the downsampled images. We use Laplacian (see Fig. 3-b) instead of grayscale images to alleviate the effect of illumination variation. In each iteration of optimization, our pyramid RGB-D odometry method minimizes the photometric error:

$$E_\text{pho} = \sum_{i=1}^{n} \left\| J_\text{r}^{[i]} \boldsymbol{\xi} - r^{[i]} \right\|^2 , \tag{5}$$

where $J_\text{r}^{[i]}$ is the Jacobian of the i-th valid pixel in the downsampled Laplacian image of R_r with respect to a 6 DoF perturbation (3 rotations and 3 translations), and $r^{[i]}$ is the residual between each valid pixel pair in the downsampled Laplacian image of R_r and the reprojection of the downsampled Laplacian image of F_n. n is the number of valid pixel pairs that are used in each iteration. Note that we start at the 5th pyramid level (coarsest) and stop at the 3rd for computational efficiency, only estimating rotation at the 5th level. In addition to efficiency, this coarse-to-fine scheme helps handle larger motion and avoids the optimization getting stuck in wrong local minimum.

C. Semi-dense RGB-D Odometry (SRO)

Inspired by [6], our SRO algorithm only uses few pixels with high intensity gradients in the residual minimization process instead of the entire image [24, 16] for better efficiency. This simplification is based on the fact that many of the calculations on the regions with low intensity gradients do not contribute much to optimizing odometry and therefore can be discarded without losing much robustness.

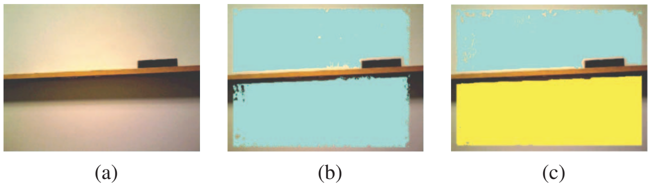


Fig. 4: (a) Color image from a keyframe shows a whiteboard offset from a wall by approximately 1cm. (b) The segmentation algorithm cannot discriminate between the wall and the whiteboard when performed on the raw depth map. (c) The fused depth map allows the algorithm to correctly segment the wall and whiteboard as separate planes.

Ideally we can directly select those pixels with large gradients only and ignore all others. However, if a high intensity gradient region corresponds to a discontinuous structure in the real world, the depth measurements of these pixels are often missing, and their reprojection cannot be calculated. As a result, we first choose some *interest pixels* with large gradients, and then select all the pixels with valid depth data within a small patch around each interest pixel (see Fig. 3-c) for calculation. Again, with the semi-dense pixels selected from the patches in both R_{r} and F_{n}, the relative transformation between the two frames can be found by minimizing the photometric error in Eq. 5. $J_{\text{r}}^{[i]}$ is now the Jacobian of the i-th selected pixel in the grayscale image of R_{r} with respect to the same 6 DoF perturbation, and $r^{[i]} = p_{\text{r}}^{[i]} - p_{\text{n}}^{[i']}$ is the residual between each selected pixel $p_{\text{r}}^{[i]}$ in R_{r} and corresponding selected pixel $p_{\text{n}}^{[i']}$ in F_{n}, with correspondences found projectively. n is again the number of valid pixel pairs that are used in each iteration.

D. Reference Frame Sharing

Because we share reference frames and warp input pixels toward the reference frame in each iteration of the optimization, the Jacobians of the image pyramid and the semi-dense patches in SRO are calculated only when a new reference frame is defined, which saves significant computation time compared to calculating the Jacobians in each frame. Another advantage of reference frame sharing is that there might be less drift accumulation compared to a frame-to-frame formulation. The overall dense RGB-D odometry method can operate faster than real-time (more than 50 fps) on a single CPU, which leaves plenty of time for data fusion and planar mapping in our keyframe structure.

V. Local Depth Fusion

The fast odometry provides a camera pose estimate at every frame, enabling data fusion and more precise planar mapping than would be possible without such pose estimates. The depth map M_i of a keyframe K_i is initialized simply with the depth image of K_i. Utilizing the same camera-motion criteria used to select keyframes, but with smaller thresholds, fusion frames are regularly selected to fuse depth data into the current local map M_i. The depth measurements from these fusion frames are projected into the depth image of K_i based on the odometry estimation and then fused into M_i using a running average method for each pixel. To avoid fusing incorrectly associated depth measurements, only measurements that are within a small threshold of the keyframe's corresponding depth measurement are fused.

With reasonably accurate pose estimates, the local fusion process produces a significantly smoothed model after fusing as few as 3-4 frames. The frequency of fusion frame selection may be adjusted to allow for real-time performance of the overall planar SLAM system. Fusion frames will continue to be selected and fused with M_i until a new keyframe K_{i+1} is selected, at which point planes will be segmented from M_i and a new local map M_{i+1} will be initialized.

We use the clustering algorithm described in [11] for plane segmentation. As shown in Fig. 4, the fused local maps enable more precise plane segmentations than are possible using just a single frame.

VI. Global Mapping with Planes

Although the odometry method described in section IV provides pose estimates at every frame, it is not as accurate or robust as state-of-the-art methods such as [16] due to the real-time and CPU only restrictions. Thus, the algorithm for finding plane correspondences must be robust to uncertainty in the pose estimate. Both our novel data association method for plane features and a procedure for performing global loop closure help to generate a globally consistent map. These methods are detailed in this section along with the plane extraction and representation.

A. Plane Representation

An infinite plane landmark is represented as a unit length homogeneous vector $\boldsymbol{\pi} = \begin{bmatrix} \mathbf{n}^\top & d \end{bmatrix}^\top \in \mathbb{P}^3$ in projective space in our global optimization, where $\mathbf{n}$ is the normal vector of the plane and d is its distance from the origin. By enforcing $\|\boldsymbol{\pi}\| = 1$ we parametrize planes on S^3. We deal with over-parametrization by using the same minimal representation $\boldsymbol{\omega} \in \mathbb{R}^3$ as for quaternions, exploiting the exponential map

$$\exp(\boldsymbol{\omega}) = \begin{pmatrix} \frac{1}{2} sinc\left(\frac{1}{2}\|\boldsymbol{\omega}\|\right)\boldsymbol{\omega} \\ \cos\left(\frac{1}{2}\|\boldsymbol{\omega}\|\right) \end{pmatrix} \in S^3. \tag{6}$$

for updating the plane during optimization, as discussed in more detail in [13].

B. Plane Fitting and Uncertainty Estimation

Following segmentation, a plane model is fitted to each point cluster using the linear model described in [8]: $\delta^{[i]} = au^{[i]} + bv^{[i]} + c$, where $\delta^{[i]}$ is the disparity measurement, $u^{[i]}$ and $v^{[i]}$ are the pixel coordinates, and a, b, and c are the unknown parameters that depend on both the (known) camera intrinsics as well as the parameters of the underlying plane $\boldsymbol{\pi}$. Our noise model assumes additive Gaussian noise on disparity, which leads to the standard least squares model

$$\left\| \begin{pmatrix} \delta^{[1]} \\ \delta^{[2]} \\ \vdots \\ \delta^{[n]} \end{pmatrix} - \begin{pmatrix} u^{[1]} & v^{[1]} & 1 \\ u^{[2]} & v^{[2]} & 1 \\ \vdots & \vdots & \vdots \\ u^{[n]} & v^{[n]} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right\|^2 = \|\mathbf{y} - X\boldsymbol{\beta}\|^2 . \tag{7}$$

The optimal parameters are solved for as $\boldsymbol{\beta}^* = \left(X^\top X\right)^{-1} X^\top \mathbf{y}$ which may be used to compute the optimal plane parameters $\boldsymbol{\pi}^* = T\left(\boldsymbol{\beta}^*\right)$. The explicit form of T is omitted for brevity. The covariance of the parameters $\boldsymbol{\beta}^*$ is $\Sigma_\beta = \left(X^\top X\right)^{-1}$. Σ_β is transformed to the space of the plane parameters $\boldsymbol{\pi}$ using the Jacobian of T computed numerically: $\Sigma_{\boldsymbol{\pi}} = J_T \Sigma_\beta J_T^\top$. However, since global optimization utilizes the minimal parametrization to update planes, the covariance matrix must match the dimensionality of the minimal parametrization (three). Therefore, we compute the 3×3 covariance matrix $\Sigma'_{\boldsymbol{\pi}} = J_l \Sigma_{\boldsymbol{\pi}} J_l^\top$ using the Jacobian of the log map, which is described in [13]. This is the final covariance matrix that is used in the global optimization.

C. Data Association

We implement a novel projective data association algorithm for matching planes between keyframes. Once planes are extracted from keyframe K_{i-1}, all of the landmarks seen in the previous 10 keyframes are considered candidates for data association and are projected into the frame of K_{i-1} using the globally optimized pose estimates. An exhaustive search across measurement-landmark pairs is used to find the best correspondences. Three criteria must be met in order to match a new plane measurement with a previously existing landmark. The first two criteria are commonly used in the literature: the normals must be within a small threshold of each other as well as the distances of the planes from the origin (we use 10° and 0.2m). The last criterion computes the residual of the landmark's plane model using the points from the plane measurement, normalized for the number of points. That is, for landmark Π_p and measurement c_q, we compute the cost

$$C_{pq} = \frac{\left\|\mathbf{y}_q - X_q \boldsymbol{\beta}_p\right\|^2}{n} \tag{8}$$

where $\mathbf{y}_q$ and X_q are the corresponding data from measurement c_q as defined in Eq. 7, n is the number of points observed in measurement c_q, and $\boldsymbol{\beta}_p$ is the vector of regression parameters corresponding to landmark Π_p. We use a threshold of 10 for C_{pq}, which was empirically determined to minimize the number of false positive correspondences while allowing for some uncertainty in the odometry motion estimate. Plane measurements from K_{i-1} that are not matched with previously observed landmarks are added to the graph as new landmarks.

D. Loop Closure

Although explicitly mapping planar features significantly helps reduce drift, the optimized trajectory is still prone to some amount of drift. Therefore, a global loop closure algorithm is necessary to correct for the drift that inevitably accumulates. Plane features are generally not descriptive enough to form the basis of either a place recognition or loop closure algorithm. We therefore utilize a bag-of-words approach to place recognition in order to register loop closure candidates [9]. Grayscale images from each keyframe are cached in a database which is queried to find matches whenever a new keyframe is added.

Upon receiving a loop closure candidate, we implement a least squares method for finding a transformation between two sets of 3-D points [2]. SURF [3] keypoints are extracted from the two keyframes in consideration, matches are found efficiently using an approximate nearest neighbor algorithm, and 3D point locations are generated by projecting the points according to the corresponding depth measurements. We utilize RANSAC to find a robust transformation between the two sets of 3D keypoints and add it to the graph as a constraint between the two keyframes. Finally, after solving the graph with the new loop closure constraint, we attempt to merge planes viewed in the two involved keyframes using the data association method presented in section VI-C. This further constrains the solution and reduces the duplication of plane landmarks.

VII. Experimental Results

A. Experimental Settings

We implement KDP-SLAM on a desktop computer with an Intel Core i7-4790 processor, and GPU being used only for visualization, not computation. There are four separate threads in the system: fast odometry estimation, local dense fusion, planar mapping (including plane extraction, association, factor graph optimization, and loop closure), and visualization. The grid size for extracting planes in IPP algorithm is 20×20, and the patch size around an interest pixel in SRO algorithm is 4×4. Our implementation, which has not been optimized yet, runs at about 30 fps in general.

We compare our KDP-SLAM to other dense RGB-D SLAM and planar SLAM methods on the synthetic ICL-NUIM datasets [10] and the real-world TUM RGB-D datasets [25] quantitatively. Additionally, we use an ASUS Xtion Pro Live to collect our own larger-scale RGB-D datasets with 30 fps and 640×480 resolution in both color and depth images. Since ground truth trajectories and maps are not available for our own sequences, we provide the 3D reconstructions generated by KDP-SLAM and Kintinuous [28] (a state-of-the-art, large-scale dense SLAM algorithm) for qualitative evaluation. ElasticFusion, which is another state-of-the-art dense SLAM system, fails catastrophically on our large-scale datasets since it is not designed for such large-scale environments.

B. Results

Table I shows the absolute trajectory (ATE) [25] root-mean-square error (RMSE) of the resulting trajectories of the *living room sequences with noise* in the ICL-NUIM synthetic dataset. See Fig. 5-a for sample 3D reconstruction using KDP-SLAM. The trajectory error of our method is comparable to the state-of-the-art, with KDP-SLAM outperforming each of the alternative methods on at least one of the sequences. Note that KDP-SLAM consistently outperforms the only other CPU-only algorithm, and occasionally outperforms the GPU-accelerated systems.

TABLE I: Comparison of ATE RMSE (unit: m) on the synthetic ICL-NUIM datasets. The *italicized* methods require GPU for computation. The errors that are smaller than ours are underlined.

System	lr kt0n	lr kt1n	lr kt2n	lr kt3n
DVO SLAM [17]	0.104	0.029	0.191	0.152
RGB-D SLAM [5]	0.026	0.008	0.018	0.433
Kintinuous [28]	0.072	0.005	0.010	0.355
ElasticFusion [29]	0.009	0.009	0.014	0.106
Dense planar SLAM [23]	0.246	0.016	-	-
CPA-SLAM [19]	0.007	0.006	0.089	0.009
KDP-SLAM (Ours)	0.009	0.019	0.029	0.153

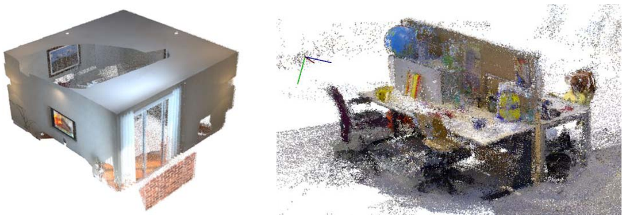

(a)
(b)

Fig. 5: KDP-SLAM reconstruction of (a) ICL-NUIM "lr kt2n" sequence and (b) TUM "freiburg3_long_office_household" sequence (10x downsampled pointcloud).

Although our system can reconstruct the general structure from the TUM datasets (see Fig. 5-b), its quantitative results (ATE RMSE) are about 5 times worse than the results of other state-of-the-art methods. This is entirely expected, as KDP-SLAM utilizes a much cheaper and less robust odometry method than those algorithms, which makes tracking difficult in the presence of strong rotation, image blur, rolling shutter effects, lighting changes or misalignment between color and depth images (all of which are present in the TUM sequences). Furthermore, many of the sequences from the TUM dataset capture highly cluttered environments with few distinguishable planes, whereas KDP-SLAM is specifically designed for highly-planar environments.

Fig. 7 shows a sample dataset gathered using a hand-held Asus Xtion Pro Live. The sequence traverses two corridors on different floors and the connecting staircases before finishing with a large loop closure. As we can observe from the results, Kintinuous distorts the planes even with loop closing, while KDP-SLAM maintains the planar structure and significantly reduces the drift in the map. More results are shown in Fig. 1 and 6 (10× downsampled pointcloud).

VIII. Discussion and Conclusion

We present a novel keyframe-based dense planar SLAM (KDP-SLAM) approach to online 3D reconstruction of indoor environments. The fast odometry method enables navigation of environments with insufficient planar constraints (such as corridors) as well as fusion of depth images into a local map, which aids the process of plane segmentation. Large-scale drift is reduced by accurately associating planes using the novel projective plane association algorithm and by incorporating explicit pose-to-pose and plane-to-plane loop closure constraints. Through experiments, we demonstrate the advantages of our system in large-scale mapping, especially in keeping the shape of planar structures from distortion, and its efficiency as a CPU-based dense RGB-D SLAM system with real-time performance.

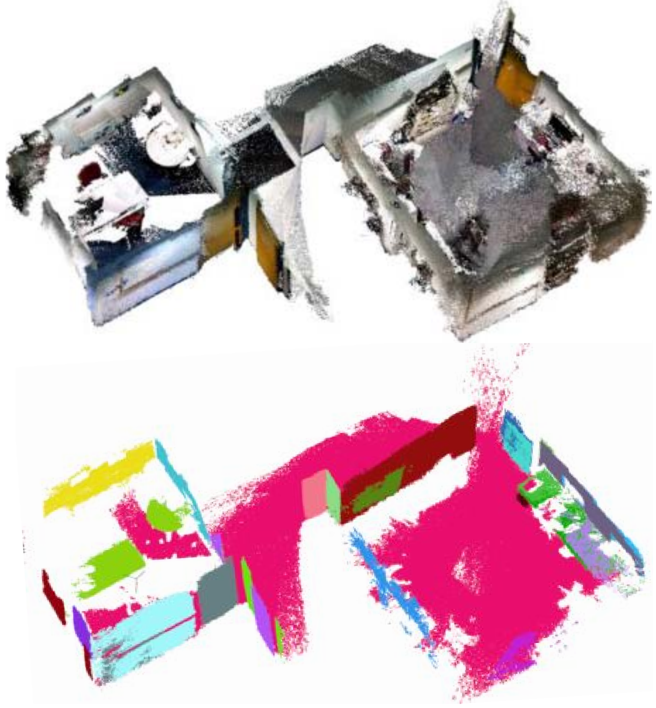


Fig. 6: Reconstruction of two rooms using our KDP-SLAM system (top) and its corresponding false-colored planar map (bottom).

There are several crucial limitations of our current system. The odometry estimates become particularly inaccurate in scenes with one or two planes and little-to-no texture. This is often encountered when turning corners in corridors with blank walls. Although the odometry tracking rarely fails catastrophically, the pose of the sensor cannot be relocalized toward the existing map or keyframes in such a case. Also, the number of keyframes can grow unbounded even if we keep mapping within the same area.

In the future, this real-time SLAM system may be improved by incorporating IMU data in the optimization in a tightly-coupled manner for robust odometry estimation. The adoption of a strategy to reuse keyframes would further enhance the large-scale applicability and allow for redundant mapping of small areas over long time-scales. The robustness of the system could also be improved by incorporating structural models of other features typical to indoor environments besides planes, such as staircases, pillars, etc. The visual quality of the map could be improved with texture mapping. Another future direction is to recover dense 3D for complex structure locally anchored to planes so that they can be optimized all together globally.

Acknowledgements

We would like to thank Thomas Whelan for his help with Kintinuous.

References

[1] Immersive 3D spaces for real-world applications | Matterport. [Online]. Available: https://matterport.com/

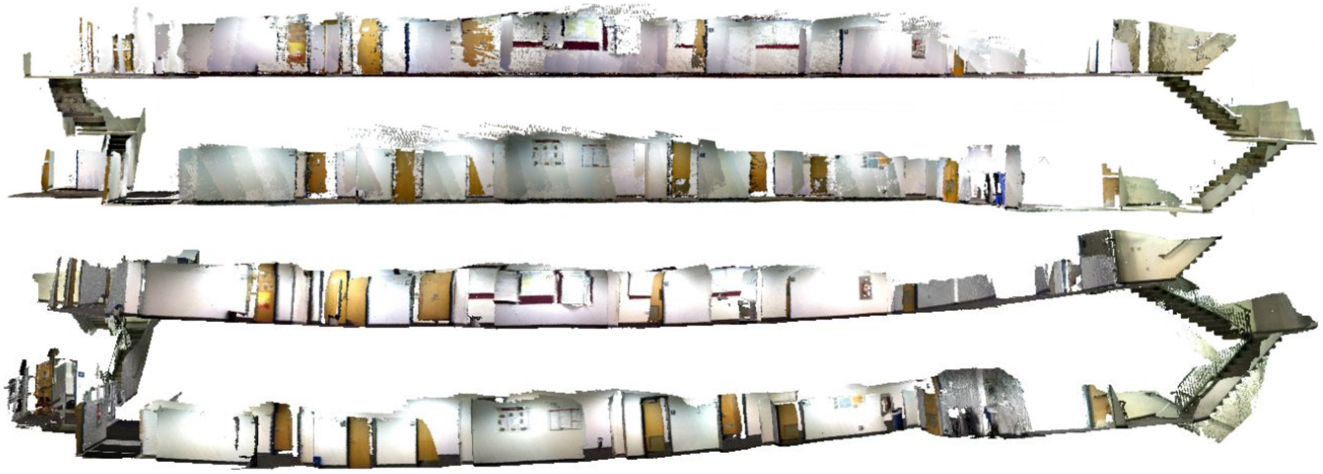


Fig. 7: Two floor dataset reconstructions by KDP-SLAM (top) and Kintinuous (bottom). Note that KDP-SLAM has reduced drift and maintained the planar structure after loop closure compared to Kintinuous.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sept 1987.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[4] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[5] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2012, pp. 1691–1696.

[6] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Intl. Conf. on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Eur. Conf. on Computer Vision (ECCV)*, September 2014.

[8] C. Erdogan, M. Paluri, and F. Dellaert, "Planar segmentation of RGBD images using fast linear fitting and Markov chain Monte Carlo," in *Conf. on Computer and Robot Vision*, May 2012, pp. 32–39.

[9] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

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

[11] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *RoboCup 2011: Robot Soccer World Cup XV*. Springer, 2012, pp. 306–317.

[12] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Intl. Symp. on Robotics Research (ISRR)*, vol. 2, 2011.

[13] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2015, pp. 4605–4611.

[14] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[15] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *Intl. Conf. on 3D Vision (3DV)*, June 2013, pp. 1–8.

[16] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2013.

[17] ——, "Dense visual SLAM for RGB-D cameras," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[18] T. Lee, S. Lim, S. Lee, S. An, and S. Oh, "Indoor mapping using planes extracted from noisy RGB-D sensors," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2012, pp. 1727–1733.

[19] L. Ma, C. Kerl, J. Stueckler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2016.

[20] R. Mur-Artal, J. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[21] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Oct 2011, pp. 127–136.

[22] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger, and A. Birk, "Fast 3D mapping by matching planes extracted from range sensor point-clouds," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2009, pp. 1150–1155.

[23] R. Salas-Moreno, B. Glocken, P. Kelly, and A. Davison, "Dense planar SLAM," in *IEEE Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Sept 2014, pp. 157–164.

[24] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Intl. Conf. on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, pp. 719–722.

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2012, pp. 573–580.

[26] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2013, pp. 5182–5189.

[27] A. Trevor, J. Rogers, and H. Christensen, "Planar surface SLAM with 3D and 2D sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2012, pp. 3041–3048.

[28] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald, "Real-time large scale dense RGB-D SLAM with volumetric fusion," *Intl. J. of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, Apr 2015.

[29] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.