Tracking an RGB-D Camera Using Points and Planes

Esra Ataer-Cansizoglu
Northeastern University
Boston, MA, USA
ataer@ece.neu.edu

Yuichi Taguchi, Srikumar Ramalingam, Tyler Garaas
Mitsubishi Electric Research Labs (MERL)
Cambridge, MA, USA
{taguchi,ramalingam,garaas}@merl.com

Abstract

Planes are dominant in most indoor and outdoor scenes and the development of a hybrid algorithm that incorporates both point and plane features provides numerous advantages. In this regard, we present a tracking algorithm for RGB-D cameras using both points and planes as primitives. We show how to extend the standard prediction-and-correction framework to include planes in addition to points. By fitting planes, we implicitly take care of the noise in the depth data that is typical in many commercially available 3D sensors. In comparison with the techniques that use only points, our tracking algorithm has fewer failure modes, and our reconstructed model is compact and more accurate. The tracking algorithm is supported by relocalization and bundle adjustment processes to demonstrate a real-time simultaneous localization and mapping (SLAM) system using a hand-held or robot-mounted RGB-D camera. Our experiments show large-scale indoor reconstruction results as point-based and plane-based 3D models, and demonstrate an improvement over the point-based tracking algorithms using a benchmark for RGB-D cameras.

1. Introduction

There has been significant progress in 3D reconstruction algorithms that simultaneously track the pose of a camera. Such techniques are commonly referred to as simultaneous localization and mapping (SLAM) and widely studied in computer vision, robotics, and augmented reality. In particular, real-time SLAM systems using a single 2D camera or RGB-D camera such as Kinect are attractive, as they provide cost-effective and easy-to-use solutions for several interactive computer vision applications.

Although many promising tracking and 3D reconstruction results have been shown, there are several challenges that still exist. SLAM systems using a single 2D camera [7, 11, 12, 14] are generally successful for textured scenes, but encounter many failure modes with texture-less regions. Systems using a depth camera [18, 16, 13, 17, 29, 5] exploit geometric variations in the scene such as curved surfaces and depth boundaries with the help of iterative-closest point (ICP) algorithms [3]. However, such ICP-based approaches fail if there are no geometric variations. Typically room-scale scenes have many objects that have both texture and geometric features. For reconstructing larger-scale scenes, we need to track features in regions like corridors with limited texture and insufficient geometric variations. Systems using an RGB-D camera [8, 9, 20, 1, 26, 10, 28] exploit both texture and geometric features to handle the problem, but they still treat the scene as a set of points and do not exploit the *structure* of the scene.

We present a SLAM system that exploits common structures of man-made scenes, i.e., planes, in conjunction with points as primitives. The goal of our system design is to enable compact representations of scenes and accurate registration that minimizes failure cases due to insufficient texture and geometric features. Figure 1 shows a motivating example, where a large indoor scene including a kitchen and several cubicles is represented by using only 15 planes. Note that the entire floor plane is modeled with a single plane, which achieves a compact representation of the scene as well as drift-free registration along the direction of the normal of the plane. In this paper, we demonstrate how to find point and plane correspondences using camera motion prediction, and develop a tracker based on a prediction-and-correction framework. We also incorporate relocalization and bundle adjustment processes using both points and planes [23] to recover from tracking failures and to continuously refine camera pose estimates. We show experimental results on large-scale indoor 3D reconstruction using a hand-held or robot-mounted Kinect sensor. We also evaluate the performance of our SLAM system using benchmark datasets [22].

Terminologies: As commonly used in the literature [11], we use *tracking* to refer to a process that uses the predicted motion of a camera for sequentially estimating the camera pose, and *relocalization* to refer to a process that uses some feature-based global registration for recovering from tracking failures.

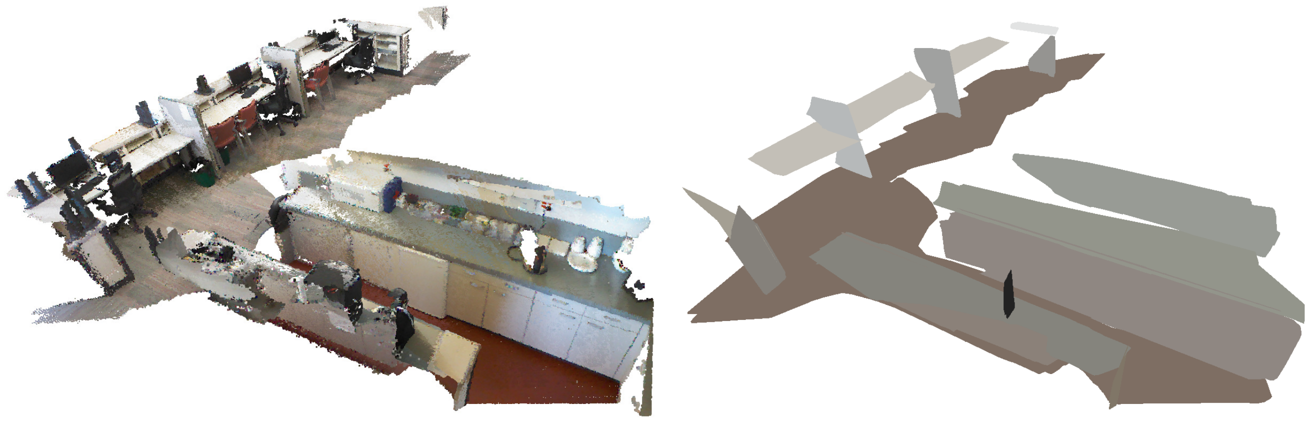


Figure 1. Man-made scenes include many planar structures. We exploit them along with point features to robustly track an RGB-D camera. In addition to registered 3D point clouds (left), our system reconstructs a large-scale indoor scene as a compact plane-based model (right). The number of keyframes registered in this model is 86, and the numbers of plane and point landmarks are 15 and 12558, respectively. Note that the entire floor is represented by a single plane. One of the major advantages of our approach is the compact modeling. The point-based representation on the other hand has an order of $86 \times 640 \times 480$ primitives.

1.1. Contributions

The following list summarizes our contributions.

- We present a prediction-based camera tracking algorithm using both points and planes.
- We describe a real-time SLAM system using the tracking algorithm as well as relocalization and bundle adjustment processes, all using points and planes.
- We demonstrate large-scale indoor scene reconstruction using a hand-held and robot-mounted RGB-D sensor.

1.2. Related Work

Camera Tracking: For systems using 3D sensors providing 3D point clouds, the camera tracking problem reduces to the registration problem given some 3D correspondences. The ICP algorithm [3] finds point-to-point or point-to-plane correspondences iteratively, starting from an initial pose estimate given by camera motion prediction. It has been widely used for line-scan 3D sensors in mobile robotics (also known as scan matching) [24] as well as for depth cameras and 3D sensors producing full 3D point clouds [18, 16]. KinectFusion [13] extended commonly used frame-to-frame ICP registration to frame-to-model ICP registration: The system accumulated registered point clouds in a truncated signed distance function (TSDF) volume [6] and generated a virtual depth map, which has much less noise, to be registered to the current depth map. It was extended for reconstructing larger-scale scenes by spatially moving the TSDF volume [17, 29] and by using a hierarchical data structure that efficiently represents empty spaces and enables lossless streaming of the data between the GPU and CPU [5]. These ICP-based methods require scenes to have sufficient geometric variations for accurate registration.

Henry et al.'s RGB-D mapping system [8] extracted features from RGB images and performed descriptor-based point matching to compute point-to-point correspondences and estimate the camera pose, which was then refined with the ICP algorithm. Instead of using a sparse set of feature points, several algorithms aligned a dense set of points between two frames by warping the points from one frame to the other using the depth map and minimizing the photometric error [20, 1, 10]. In addition to the photometric error, the depth differences between points [26] and the point-plane ICP cost [28] were also incorporated into the direct alignment framework. Those systems exploit both texture (RGB) and geometric (depth) features in the scene; however, they still use a set of points as primitives, which results in redundant representations of scenes and causes more drifts than systems using higher-level primitives.

SLAM Using High-Level Primitives: Plane features have been used in several SLAM systems [27, 15, 25]. To determine the camera pose, we need at least three planes whose normals span $\mathbb{R}^3$; thus, using only planes causes many degeneracy issues especially if the field of view (FOV) or range of the sensor is small such as in Kinect. Trevor et al. [25] used a large FOV line-scan 3D sensor along with a small FOV depth camera to avoid the degeneracy with an additional system cost. Salas-Moreno et al. [19] used objects as primitives in their SLAM system. Although objects are more distinctive and provide more constraints to camera poses than planes, objects are not as general as planes in typical scenes and their models need to be prescanned.

Taguchi et al. [23] presented point-plane SLAM, which uses both points and planes to avoid the failure modes

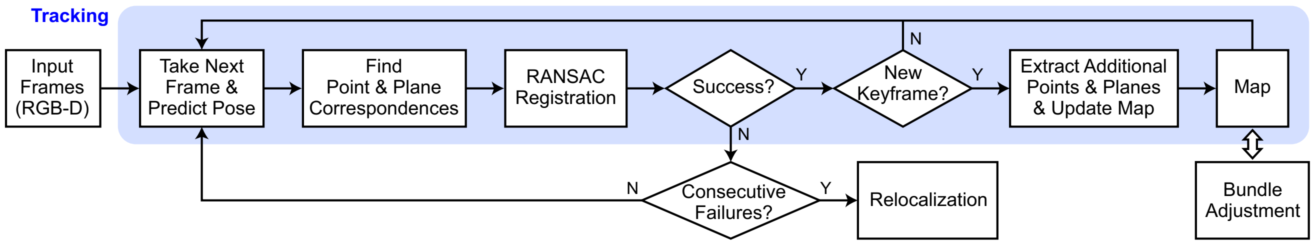


Figure 2. System overview. The system performs prediction-based tracking using points and planes to determine the pose of an RGB-D camera. If the tracking fails in consecutive frames, the system uses a relocalization process to recover the pose. The system also runs bundle adjustment to refine the poses of keyframes and point and plane landmarks in the map in an asynchronous thread.

that are common in algorithms using one of these primitives. The system did not use any camera motion prediction; instead it performed relocalization for all the frames by finding point and plane correspondences globally. As a result, the system was slow (2–3 frames per second) and encountered failures with some repetitive textures due to descriptor-based point matching. Our prediction-based tracking achieves faster correspondence search and shows robustness in scenes with repetitive textures.

2. System Overview

Figure 2 shows an overview of our system, which takes a sequence of RGB-D frames as the input. Our system is a keyframe-based SLAM system [11, 21], where we select several representative frames as *keyframes* and store them in a global *map*. We use both points and planes as primitives in all the processes in the system. Points and planes in each frame are called *measurements*, and measurements in the keyframes are stored in the map as *landmarks*.

Given the map, we use a prediction-and-correction framework to estimate the pose of the current frame: We first predict the pose of the frame and use it to determine correspondences between point/plane measurements and point/plane landmarks, which are then used to correct the pose. Section 3 details the tracking algorithm.

Tracking may fail due to incorrect or insufficient correspondences. Once we observe consecutive tracking failures, we resort to the relocalization process, where we use global point and plane correspondence search between the current frame and the map, similar to [23]. We also perform bundle adjustment using points and planes [23] to refine landmarks in the map asynchronously in a separate thread.

Note that such a combination of tracking, relocalization, and bundle adjustment is common in practical SLAM systems [11, 30]. This paper advances such systems, which are generally based on points, by additionally using planes for efficient and robust SLAM.

3. Camera Pose Tracking

This section describes our tracking algorithm using features that include both points and planes. The algorithm is based on a prediction-and-correction scheme and is summarized as follows:

1. For every new RGB-D frame, predict its pose using a camera motion model.
2. Based on the predicted pose, find point and plane measurements in the new frame corresponding to point and plane landmarks in the map.
3. Perform a RANSAC-based registration using the point and plane correspondences.
4. If the pose is novel compared to those of the existing keyframes, then extract additional point and plane measurements and add the frame as a new keyframe to the map.

The next subsections discuss the steps in detail.

3.1. Camera Motion Prediction

We represent the pose of the kth frame as

$$\mathsf{T}_k = \begin{pmatrix} \mathsf{R}_k & \mathbf{t}_k \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}, \tag{1}$$

where R_k and $\mathbf{t}_k$ respectively denote the rotation matrix and the translation vector. We define the coordinate system of the map using the first frame; thus T_1 is the identity matrix and T_k represents the pose of the kth frame with respect to the map.

We predict the pose of the kth frame, $\hat{\mathsf{T}}_k$, by simply using the constant velocity assumption. Let $\Delta\mathsf{T}$ denote the previously estimated motion between the $(k-1)$th frame and $(k-2)$th frame, i.e., $\Delta\mathsf{T} = \mathsf{T}_{k-1}{\mathsf{T}_{k-2}}^{-1}$. Then we predict the pose of the kth frame as $\hat{\mathsf{T}}_k = \Delta\mathsf{T}\mathsf{T}_{k-1}$.

3.2. Finding Point and Plane Correspondences

Using the predicted pose $\hat{\mathsf{T}}_k$, we find point and plane measurements in the kth frame corresponding to landmarks in the map, as illustrated in Figure 3.

Point Correspondence: Let $\mathbf{p}_i = (x_i, y_i, z_i, 1)^\mathsf{T}$ denote the ith point landmark in the map, represented as a homogeneous vector. The 2D image projection of $\mathbf{p}_i$ in the current frame is predicted as

$$\hat{\mathbf{p}}_i^k = \hat{\mathsf{T}}_k \mathbf{p}_i, \quad \hat{\mathbf{u}}_i^k = \mathtt{FP}(\hat{\mathbf{p}}_i^k), \tag{2}$$

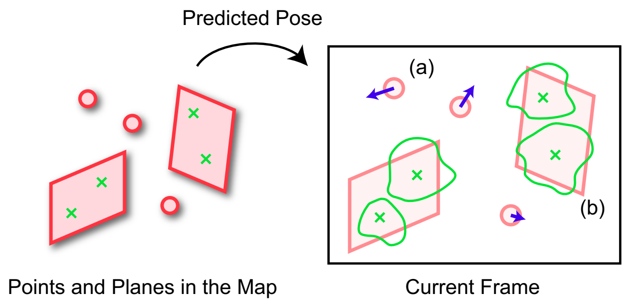


Figure 3. Given the predicted pose of the current frame, we find correspondences between point/plane landmarks in the map and point/plane measurements in the current frame. We first transform the landmarks in the map to the current frame using the predicted pose. Then, (a) for every point, we perform local search using an optical flow method from the predicted pixel location in the current frame; (b) for every plane, we first find the parameters of the predicted plane. We then consider a set of reference points on the predicted plane, and find pixels connected from each reference point that lie on the predicted plane. The reference point with the largest number of connected pixels is chosen and the plane parameters are refined using all the connected pixels.

where $\hat{\mathbf{p}}_i^k$ is the 3D point transformed to the coordinate system of the kth frame, and the function $\mathtt{FP}(\cdot)$ computes the forward projection of the 3D point onto the image plane using the internal camera calibration parameters. We find the corresponding point measurement by using Lucas-Kanade's optical flow method [4], starting from the initial position of $\hat{\mathbf{u}}_i^k$. Let $\Delta\mathbf{u}_i^k$ be the computed optical flow vector. Then the corresponding point measurement $\mathbf{p}_i^k$ is given as

$$\mathbf{u}_i^k = \hat{\mathbf{u}}_i^k + \Delta\mathbf{u}_i^k, \quad \mathbf{p}_i^k = \mathtt{BP}(\mathbf{u}_i^k)\mathrm{D}(\mathbf{u}_i^k), \tag{3}$$

where the function $\mathtt{BP}(\cdot)$ back-projects the 2D image pixel to a 3D ray and $\mathrm{D}(\cdot)$ refers to the depth value of the pixel. If the optical flow vector is not computed or the pixel location $\mathbf{u}_i^k$ has an invalid depth value[1], then the feature is regarded as lost.

Plane Correspondence: Instead of performing a time-consuming plane extraction procedure on each frame independently from other frames, as is done in existing systems [27, 15, 25, 23], we make use of the predicted pose to extract planes. This leads to faster plane measurement extraction and also provides the plane correspondences.

Let $\pi_j = (a_j, b_j, c_j, d_j)^\mathsf{T}$ denote the plane equation of the jth plane landmark in the map. We assume that the plane landmark and its corresponding measurement have some overlapping regions observed from the camera. To find such a corresponding plane measurement, we randomly choose several reference points $\mathbf{q}_{j,r}$ $(r = 1, \ldots, N)$ from the inliers of the jth plane landmark, and transform them to the kth frame as

$$\hat{\mathbf{q}}_{j,r}^k = \hat{\mathsf{T}}_k \mathbf{q}_{j,r}, \quad \hat{\mathbf{v}}_{j,r}^k = \mathtt{FP}(\hat{\mathbf{q}}_{j,r}^k) \quad (r = 1, \ldots, N). \tag{4}$$

We also transform π_j to the kth frame as

$$\hat{\pi}_j^k = \hat{\mathsf{T}}_k^{-\mathsf{T}} \pi_j. \tag{5}$$

For each transformed reference point $\hat{\mathbf{v}}_{j,r}^k$, we find pixels that are connected to it and lie on the plane $\hat{\pi}_j^k$. We then select the reference point with the largest number of inliers. The inliers are used to refine the plane equation, resulting in the corresponding plane measurement π_j^k. If the number of inliers is smaller than a threshold, the plane landmark is declared as lost. In experiments, we used $N = 5$ reference points, a threshold of 50 mm for the point-to-plane distance to determine inliers on a plane, and 9000 as the threshold of the minimum number of inliers.

Landmark Selection: Performing the above process using all the landmarks in the map would be inefficient. We thus use the landmarks appearing in a single keyframe that is the closest to the current frame. The closest keyframe is selected by using the pose of the previous frame T_{k-1} before the tracking process.

3.3. RANSAC Registration

The prediction-based correspondence search provides candidates of point-to-point and plane-to-plane correspondences, which may include outliers. Thus we run a RANSAC-based registration algorithm to determine inliers and compute the camera pose. We use the algorithm presented in [23], which solves the registration problem using the mixed correspondences in closed-form. The algorithm prioritizes plane correspondences over point correspondences, because the number of planes is typically much smaller than the number of points, and planes are more robust to noise due to the support from many points. Tracking is considered successful if the RANSAC algorithm finds a sufficient number of inliers (40% of the number of all point and plane measurements in experiments). The algorithm gives the corrected pose of the kth frame, T_k.

3.4. Map Update

We determine the kth frame as a keyframe if the estimated pose T_k is sufficiently different from the poses of any existing keyframes in the map (we used thresholds of 100 mm in translation and 5° in rotation). For the new keyframe, the point and plane measurements found as inliers in the RANSAC-based registration are associated to corresponding landmarks, while those found as outliers are discarded. We then extract additional point and plane measurements, which newly appear in this frame. The additional point

[1]We consider a depth value invalid if it is missing in the sensor reading or it is larger than a maximum depth value, which we set 3000 mm in experiments. We ignore pixels with invalid depth values in all the processes in our system.

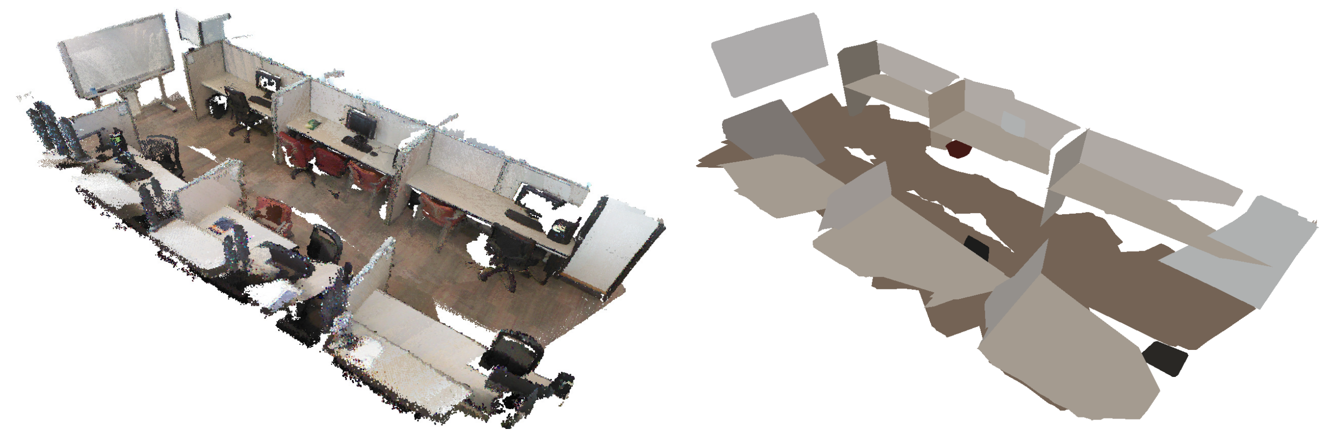
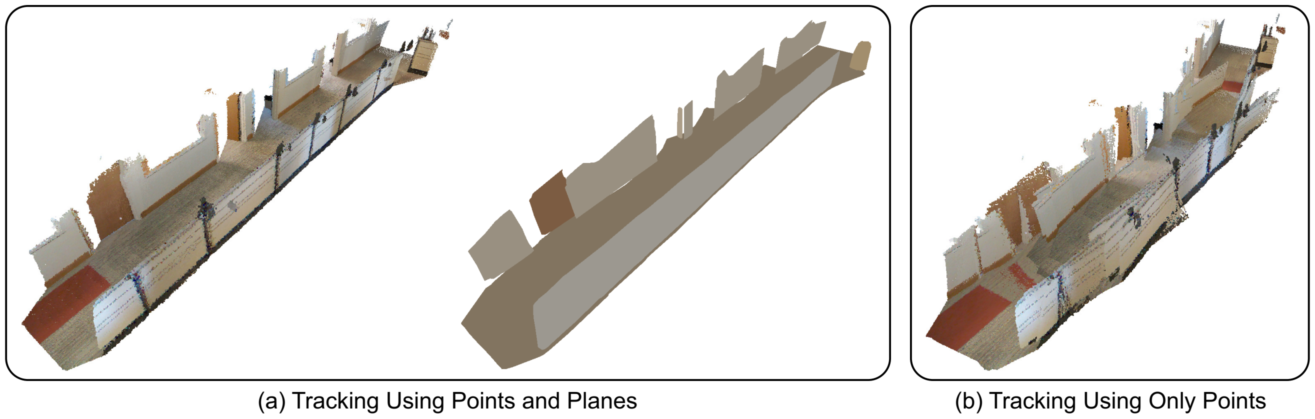


Figure 4. Office scene: Point-based (left) and plane-based (right) models of a large indoor scene with several office cubicles (124 keyframes registered using 15 plane and 15431 point landmarks). Note that the floor corresponds to a single plane landmark, and the 6 table tops are associated to a single plane landmark. These long-range interactions lead to stable, drift-less registration.


(a) Tracking Using Points and Planes
(b) Tracking Using Only Points
Figure 5. Corridor scene: (a) Models reconstructed using both points and planes for tracking (63 keyframes registered using 6 plane and 4006 point landmarks). (b) Reconstructed model using only points for tracking.

measurements are extracted using the SURF keypoint detector [2] on pixels that are not close to any existing point measurements. On the other hand, the additional plane measurements are extracted by using a RANSAC-based plane fitting algorithm [23] on pixels that are not inliers of any existing plane measurements. The additional point and plane measurements are added as new landmarks to the map. In addition, we extract SURF feature descriptors for all point measurements in the frame, which are used for relocalization.

4. Experiments

In this section, we present evaluations of our real-time tracking and SLAM system. We first demonstrate qualitative results using indoor video sequences, captured with a Kinect sensor that provides color images and depth maps at a resolution of 640×480 pixels. The sensor was either hand-held or mounted on a mobile robot. We then show quantitative analysis using two datasets from a benchmark [22].

4.1. Qualitative Results and System Performance

We tested our tracking and SLAM system for several indoor scenes. Please refer to the supplementary video, demonstrating real-time SLAM results for scenes shown in Figures 1, 4, and 5. In the video, when there is a tracking failure we indicate that by showing the entire background in green (this should not be confused with transparent green polygons indicating plane landmarks). Note that, in these sequences, our algorithm recovered from such tracking failures in the next few frames and thus did not invoke the relocalization process.

Figure 6 explains the visualization of our system. Our system provides real-time feedback to the user using this visualization, enabling the user to decide on where to scan next. If the tracking or relocalization fails, the user can return to a location from which some of the landmarks already registered in the map are observable.

Figures 1 and 4 show our reconstruction results as point-based and plane-based models. In plane-based models,

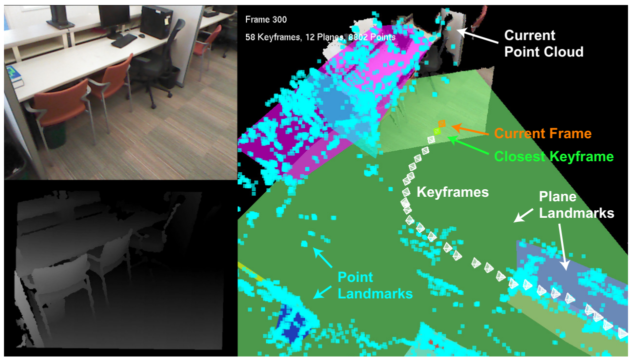


Figure 6. An example of our real-time SLAM visualization. The orange and green camera icons depict the pose of the current frame and that of the closest keyframe, respectively. The camera trajectory is shown as a set of keyframes with the white camera icons. Point landmarks (cyan points) and plane landmarks (colored polygons) are superimposed on the current point cloud.

we depict plane measurements associated to a single plane landmark using the same color (the average color of all the inlier points of the plane measurements). Note that the floor and several tables tops, each of which physically corresponds to a single plane, are successfully associated to a single plane landmark. These large planes produce long-range interactions between frames; e.g., frames observing the floor are always registered with respect to the floor plane correctly. This makes the registration accurate and drift-less.

Comparisons: Figure 5 demonstrates the advantage of our system by comparing (a) the result obtained using both point and plane correspondences with (b) that obtained using only point correspondences on a long corridor sequence. The sequence was captured using a Kinect sensor mounted on a mobile robot moving along a corridor. Most of the frames in the sequence have the floor, left and right walls as planes, and only a small number of keypoints. Because of the small number of keypoints and poor texture information, tracking using only points caused inaccurate registration and reconstruction as shown in Figure 5 (b). The floor and wall planes provide the necessary long-range constraints in this sequence; our system exploiting them produced the correct reconstruction result shown in Figure 5 (a).

Note that using only plane correspondences would fail on this sequence, because there are only two non-degenerate planes and we cannot compute the remaining 1 degree-of-freedom (along the direction of the corridor) from the planes. ICP-based approaches [16, 13, 17, 29, 5] would also drift due to insufficient geometric variations along this direction. Whelan et al. [28] combined the ICP-based camera tracking with other algorithms aligning dense textures [20] and matching sparse texture features [9], and demonstrated successful results for corridor sequences similar to Figure 5. Their system still represents a scene as a set of points (or a mesh model generated from them), while ours exploits planar structures in the scene explicitly for more compact representation.

Table 1. Average processing time of each component in our system. The tracking is performed on every frame, while the tracking with map update is performed only on keyframes.

Process	Time (msec)
Point Correspondence	4
Plane Correspondence	97
RANSAC Registration	1
Tracking Total	**102**
Additional Point Detection	53
Additional Plane Detection	132
Descriptor Extraction	15
Map Update	4
Tracking with Map Update Total	**306**

Processing Time: Table 1 reports the processing time for each component of our system, measured on an Intel Core i7-2600 PC and averaged over the corridor sequence shown in Figure 5. Currently our system runs at 10 frames per second for tracking without map update (frames that are not added to the map) and 3 frames per second for tracking with map update[2]. The tracking is more than 3 times faster than the relocalization-based system presented in [23]. As seen from the table, the tracking process allows us to avoid unnecessary descriptor extraction and matching for every frame. Moreover, the new planes are extracted only for the keyframes and this extraction is done only for a partial set of pixels that are not the inliers of the existing planes.

4.2. Quantitative Results on Benchmark Datasets

In the second set of experiments, we quantitatively evaluate the performance of our system using two sequences, `freiburg1_floor` and `freiburg1_xyz`, from a benchmark [22]. We use the absolute trajectory error (ATE) and relative pose error (RPE) proposed in [22] as performance metrics. ATE measures the distances between camera centers of the true and estimated trajectories after a global alignment of the two trajectories. In contrast, RPE computes the relative pose errors of all pairs of poses in a fixed time interval (we used the default time window of 1 second). As a result, ATE accounts for the global consistency of the trajectory, while RPE measures the registration accuracy locally over a fixed time interval.

Table 2 shows the root mean square (RMS) of each error metric for the two sequences, and Figure 7 shows the point-based and plane-based models reconstructed by our tracking-based SLAM system. We compare our tracking-based SLAM system using both points and planes with that

[2]In the supplementary video, we play back all frames with a constant speed.

Table 2. Quantitative evaluations for the `freiburg1_floor` (top) and `freiburg1_xyz` (bottom) sequences from the benchmark [22]. From left to right, the columns show the number of successfully registered frames in the sequence, the number of relocalization occurred (after tracking failures), the number of keyframes added to the map, the total number of frames, and the root mean square (RMS) of absolute trajectory error (ATE) and relative pose error (RPE) in terms of translation and rotation. The first sequence mainly consists of a large floor plane having similar or repetitive textures, which greatly benefits from our tracking-based system using both points and planes. In contrast, the second sequence includes a texture-rich scene, where the performance of the point-plane tracking is similar to that of the point-only tracking.

Method	# Success	# Relocalization	# Keyframes	# Frames	RMS of ATE	RMS of RPE
Tracking-Based (Point-Plane)	830	1	81	1223	62 mm	35 mm, 2.2°
Tracking-Based (Point Only)	558	6	60	1223	162 mm	63 mm, 1.5°
Relocalization-Based (Point-Plane) [23]	705	N/A	68	1223	65 mm	53 mm, 3.2°

Method	# Success	# Relocalization	# Keyframes	# Frames	RMS of ATE	RMS of RPE
Tracking-Based (Point-Plane)	775	2	29	787	32 mm	36 mm, 2.3°
Tracking-Based (Point Only)	749	4	34	787	26 mm	44 mm, 2.2°
Relocalization-Based (Point-Plane) [23]	714	N/A	33	787	24 mm	34 mm, 1.7°
Relocalization-Based (Point Only)	584	N/A	26	787	23 mm	30 mm, 1.2°

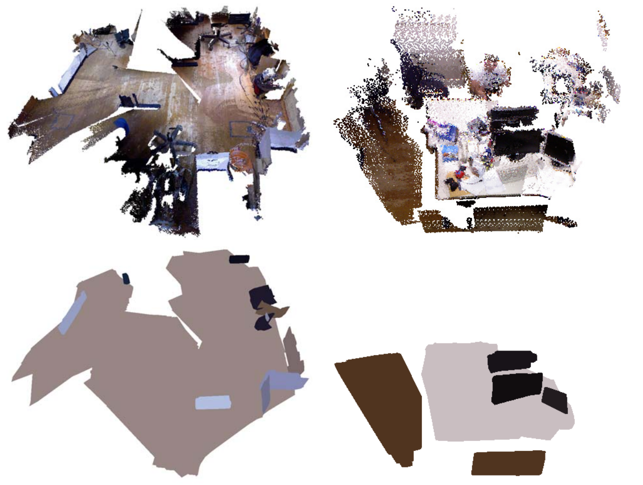


Figure 7. Point-based (top) and plane-based (bottom) models reconstructed by our system for the `freiburg1_floor` (left) and `freiburg1_xyz` (right) sequences in the benchmark [22].

using only points. In addition, we run the relocalization-based SLAM system using both points and planes as presented in [23], and that using only points. Note that we only include the keyframes in the estimated trajectory for the evaluation, since the remaining frames are not added to the map.

The first sequence (`freiburg1_floor`) contains an office floor with several knotholes on it. The results are shown in Table 2 (top). Using only points in the tracking-based system gives poor results in terms of accuracy compared to using both planes and points. This is expected, because the large floor plane is visible in most of the frames throughout the sequence and its correspondence provides more robust registration, similar to the corridor sequence in Figure 5. The tracking-based system using only points fails when the image has less texture information. Compared to the relocalization-based SLAM system using both points and planes [23], our tracking-based system provides a larger number of frames that are successfully registered. This is mainly due to the fact that the floor has repetitive textures, which leads to the failure of descriptor-based point matching algorithms. By tracking the point features, our system provides robustness to such repetitive textures. The relocalization-based SLAM system using only points failed after the first 55 frames, so we omitted its result in the table.

The second sequence (`freiburg1_xyz`) includes an office desk with several objects on it. This sequence is especially good for point-based systems, because it contains rich texture information in every frame. In this sequence, our approach is comparable to that using only points, as can be seen in Table 2 (bottom).

5. Conclusions and Discussion

We presented a tracking algorithm that exploits both point and plane correspondences for accurate and robust registration while minimizing failure cases. We developed a real-time SLAM system using the tracking algorithm in conjunction with relocalization and bundle adjustment processes. The tracking framework accelerated the feature detection and correspondence search, and also enabled us to avoid incorrect correspondences in areas with repetitive texture. We performed several qualitative and quantitative evaluations on indoor scenes and showed that planes providing long-range interactions lead to accurate and drift-less registration. Nevertheless, there are still some drift along the directions not supported by planes. For example, if the scene includes a floor plane only, registration error similar to point-based algorithms can happen on the in-plane translations and rotation (3 degrees-of-freedom). Planes provide the effect of loop closing due to long-range interactions, but we currently do not perform explicit loop closing for points.

We plan to explore it to further improve registration accuracy in the future.

Acknowledgments: We thank Jay Thornton, Jeroen van Baar, Yong-Dian Jian, and Chen Feng for valuable discussion. This work was supported by and done at MERL.

References

[1] C. Audras, A. I. Comport, M. Meilland, and P. Rives. Real-time dense appearance-based SLAM for RGB-D sensors. In *Proc. Australian Conf. Robotics and Automation*, Dec. 2011.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, Feb. 1992.

[4] J.-Y. Bouguet. Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm. Technical report, Intel Corporation, 2001.

[5] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graphics*, 32(4):113:1–113:16, July 2013.

[6] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH*, pages 303–312, Aug. 1996.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. Int'l Symp. Experimental Robotics (ISER)*, Dec. 2010.

[9] 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 *Proc. Int'l Symp. Robotics Research (ISRR)*, Aug. 2011.

[10] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2013.

[11] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, pages 1–10, Nov. 2007.

[12] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505, June 2010.

[13] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, pages 127–136, Oct. 2011.

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2320–2327, Nov. 2011.

[15] K. Pathak, A. Birk, N. Vaškevičius, and J. Poppinga. Fast registration based on noisy planes with unknown correspondences for 3-D mapping. *IEEE Trans. Robotics*, 26(3):424–441, June 2010.

[16] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a depth camera: Parameter exploration for fast ICP. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 3824–3829, Sept. 2011.

[17] H. Roth and M. Vona. Moving volume KinectFusion. In *Proc. British Machine Vision Conf. (BMVC)*, Sept. 2012.

[18] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Trans. Graphics*, 21(3):438–446, July 2002.

[19] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[20] F. Steinbrücker, J. Sturm, and D. Cremers. Real-time visual odometry from dense RGB-D images. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV) Workshops*, pages 719–722, Nov. 2011.

[21] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 2657–2664, May 2010.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 573–580, Oct. 2012.

[23] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng. Point-plane SLAM for hand-held 3D sensors. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2013.

[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[25] A. J. B. Trevor, J. G. Rogers III, and H. I. Christensen. Planar surface SLAM with 3D and 2D sensors. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3041–3048, May 2012.

[26] T. Tykkälä, C. Audras, and A. I. Comport. Direct iterative closest point for real-time visual odometry. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV) Workshops*, pages 2050–2056, Nov. 2011.

[27] J. Weingarten and R. Siegwart. 3D SLAM using planar segments. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 3062–3067, Oct. 2006.

[28] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2013.

[29] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *Proc. RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012.

[30] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, Oct. 2007.