2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

ManhattanSLAM: Robust Planar Tracking and Mapping Leveraging Mixture of Manhattan Frames

Raza Yunus[1], Yanyan Li[1] and Federico Tombari[1,2]

***Abstract*— In this paper, a robust RGB-D SLAM system is proposed to utilize the structural information in indoor scenes, allowing for accurate tracking and efficient dense mapping on a CPU. Prior works have used the Manhattan World (MW) assumption to estimate low-drift camera pose, in turn limiting the applications of such systems. This paper, in contrast, proposes a novel approach delivering robust tracking in MW and non-MW environments. We check orthogonal relations between planes to directly detect Manhattan Frames, modeling the scene as a Mixture of Manhattan Frames. For MW scenes, we decouple pose estimation and provide a novel drift-free rotation estimation based on Manhattan Frame observations. For translation estimation in MW scenes and full camera pose estimation in non-MW scenes, we make use of point, line and plane features for robust tracking in challenging scenes. Additionally, by exploiting plane features detected in each frame, we also propose an efficient surfel-based dense mapping strategy, which divides each image into planar and non-planar regions. Planar surfels are initialized directly from sparse planes in our map while non-planar surfels are built by extracting superpixels. We evaluate our method on public benchmarks for pose estimation, drift and reconstruction accuracy, achieving superior performance compared to other state-of-the-art methods. We will open-source our code in the future.**

I. Introduction

Environment-agnostic tracking and mapping based on an RGB-D camera play a central role in robotic and mixed/augmented reality applications. Such systems enable various interactive tasks, relying on accurate pose estimates and dense reconstruction.

Among traditional methods, feature-based approaches are more robust to illumination changes, compared to direct methods. Pure point-based methods [1], [2] lead to unstable performance in low-textured scenes. Robustness can be improved by adding other geometric features, like lines and planes, to the system [3], [4], [5].

Without the use of global bundle adjustment and loop closure [2], [6], small errors in pose estimation accumulate over time, causing drift in the camera pose trajectory. The former is computationally expensive, especially with large maps, and the latter works only if the agent revisits a location. Another approach for drift reduction is the use of the Manhattan/Atlanta World assumption to estimate rotation, given the fact that drift is mostly driven by inaccurate rotation estimations [7], [8]. This technique has been employed by [9], [10] and our previous works [11], [12], which exploit parallel and perpendicular relationships between geometric features in the scene. These methods model the environment as a single global Manhattan World (MW) and make the assumption for every frame, which is very limiting.

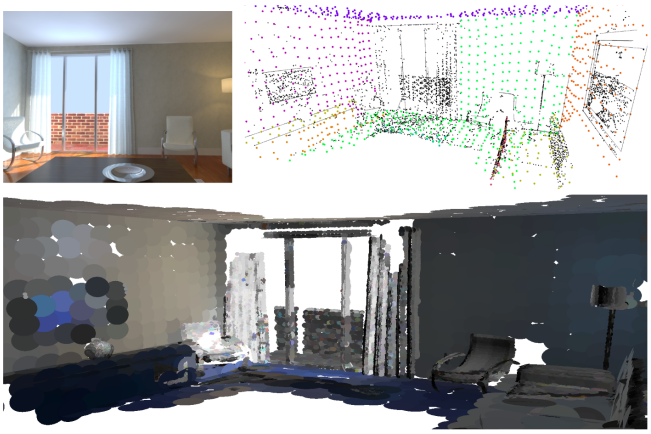

Fig. 1. Indoor reconstruction from the proposed ManhattanSLAM framework. **Top Left:** Sample indoor scene. **Top Right:** Sparse reconstruction of the indoor environment. **Bottom:** Dense surfel-based reconstruction of the same environment.

In this paper, we alleviate the stringent requirement of the MW assumption by proposing a framework which can robustly utilize the MW structure, while also working in non-MW scenes, using point-line-plane tracking. We provide a method to detect Manhattan Frames (MF) directly from planes, allowing us to model the scene as a Mixture of Manhattan Frames (MMF) [13], which is more applicable to real-world scenes, and estimate drift-free rotation by tracking MF observations across frames. Moreover, if no MFs are detected, our method switches to feature tracking, thus making it more robust than existing MW-based methods, as shown by our evaluation.

Additionally, to provide a dense map for robots, we propose an efficient surfel-based dense mapping strategy based on [14]. Different from [15] and [14], where surfels are created for every pixel or superpixel, our method divides each keyframe into planar and non-planar regions. Surfels are initialized either from superpixels for non-planar regions or directly from our sparse map plane points for planar regions. Therefore, compared to prior methods, the proposed strategy provides a more memory efficient dense reconstruction. The main contributions of this paper are summarized as:

i) A robust and general SLAM framework for indoor scenes, which takes the best of both worlds (MW assumption and feature tracking) by relying, when possible, on the MW structure for drift-free rotation estimation but able to

[1]:Technical University of Munich, Germany; {raza.yunus, yanyan.li, federico.tombari}@tum.de [2]:Google Inc.

We gratefully acknowledge Stefano Gasperini for the helpful discussion. Yanyan Li is the corresponding author.

seamlessly switch to feature tracking when MW does not hold.

ii) A novel drift-free rotation estimation method that tracks MF observations with the help of a Manhattan Map, generated by a suitable MF detection approach.

iii) An efficient dense surfel-based mapping strategy, which represents non-planar and planar regions by using superpixels and sparse plane points, respectively.

II. Related Work

ORB-SLAM [2] is a popular point-based monocular SLAM system, which extends the multi-threaded and keyframe-based architecture of PTAM [1]. It uses ORB features, builds a co-visibility graph and performs loop closing and relocalization tasks. ORB-SLAM2 [6] further extends it to stereo and RGB-D sensors, while in ORB-SLAM3 [16], inertial data, a multi-map module and support for an abstract camera model are integrated into the system. To improve the robustness of point-based methods, lines and planes are extracted from the environment to deal with low/non-textured scenes. [17] and [18] are extended from EKF-SLAM, building 3D line-based maps. [19] constructs a 3D line-based SLAM system using Plücker line coordinates for initialization and projection of 3D lines, and a 4 DoF orthonormal representation for optimization. Moreover, two recent homonymous techniques were proposed, PL-SLAM [3], [20], which merge lines into a point-based system. [21] provides a RANSAC-based registration method for localization with hand-held 3D sensors, registering using points, planes, or a combination of them. In CPA-SLAM [22], direct and dense DVO-SLAM [23] is extended to incorporate global plane landmarks for tracking the pose, in an Expectation-Maximization framework. [24] models infinite planes using the homogeneous parametrization and provides a minimum representation of planes for optimization, i.e., its azimuth angle, elevation angle and distance from the origin. Inspired by the MW assumption, SP-SLAM [5] adds constraints between parallel and perpendicular planes in the scene.

Based on the MW assumption, [8] proposes a mean-shift algorithm to track the rotation of MF across scenes, while using 1-D density alignments for translation estimation. OPVO [9] improves the translation estimation by using the KLT tracker. Both methods require two planes to be visible in the frame at all times. LPVO [25] eases this requirement by incorporating lines into the system. Structural lines are aligned with the axes of the dominant MF and can be integrated into the mean shift algorithm, improving robustness. Hence for LPVO, only a single plane is required to be visible in the scene, given the presence of lines. Drift can still occur in translation estimation as it relies on frame-to-frame tracking. To tackle this, L-SLAM [10] adds orthogonal plane detection and tracking on top of the LPVO architecture in a filtering framework. [26] extends the mean-shift algorithm for the more general scenario of Atlanta World, which can represent a wider range of scenes. [11] allows the use of mean-shift algorithm for monocular scenes, by estimating surface normals from an RGB image using a convolutional neural network. [12] further improves translation estimation by tracking plane features, in addition to points and lines, and adding parallel and perpendicular constraints between them.

KinectFusion [27] provides an algorithm for real-time dense mapping and tracking of surfaces on a GPU using ICP alignment and a volumetric TSDF model. ElasticFusion [15] is another GPU-based approach that provides surfel-based maps of the environment. [28] builds a dense map using surfels, grouping them in large areas with little or no curvature to form planar regions that provide a semantic value for different applications. BundleFusion [29] builds a globally consistent 3D reconstruction using a sparse-to-dense alignment strategy. Recently, real-time GPU-based mesh reconstruction techniques are proposed in [30] and [31]. [14] proposes superpixel-based surfels to decrease the number of surfels in the map, which enables them to run their implementation on a CPU.

III. Method

Our system tackles three main tasks: tracking, sparse mapping and dense mapping, as shown in Figure 2. In this section, we explain the essential components of our system, including our novel approach for MF detection, drift-free rotation estimation and dense mapping.

A. Tracking Framework

For each RGB-D frame, points and lines are extracted from the RGB image while planes are extracted from the depth map. Similar to [6], we make use of the constant velocity motion model to get an initial pose estimate, that is further optimized by feature correspondences and structural regularities. For points and lines, a guided search from the last frame is used to match features, and planes are matched directly in the global map. Then, we detect MFs to determine whether the current scene is an MW scene or a non-MW scene, using the respective strategies for pose estimation, as described in Section III-D. As an additional step in both cases, we track features in the local map of the current frame to further refine pose estimates. A new keyframe is created if the current frame observes less than 90% of the points observed in the previous frame.

B. Feature Detection and Matching

Since points are difficult to extract in low-textured scenes, the proposed system also exploits the structural information of the environment, through lines and planes.

1) Points: For points, we use ORB features, which are based on the FAST keypoint detector [32] and BRIEF descriptor [33]. A 3D point is represented as $P = (X, Y, Z)$, while its 2D observation is represented as $p_{obs} = (u, v)$. Matches are determined by projecting 3D points on the image and finding the closest observation using Hamming distance between the respective descriptors.

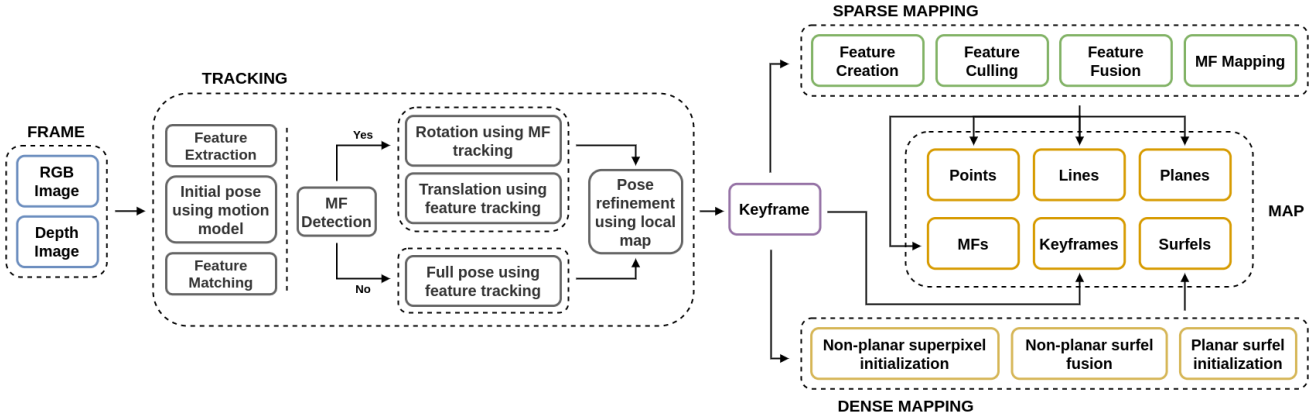

Fig. 2. Overview of the system showing three main tasks, i.e., tracking, sparse mapping and dense mapping, and components of the map.

2) Lines: To detect and describe line segments in the image, we use the robust LSD detector [34] and the LBD descriptor [35]. We represent 3D lines and their 2D observations with their endpoints (P^l_{start}, P^l_{end}) and (p^l_{start}, p^l_{end}) respectively while also obtaining normalized line function for the observed 2D line as $l_{obs} = p^l_{start} \times p^l_{end} / \|p^l_{start}\| \|p^l_{end}\| = (a,b,c)$. To determine a match between a 3D line and a 2D observation, both endpoints of the 3D line are individually projected and matched using the LBD descriptor.

3) Planes: Planes are extracted from the downsampled 3D point cloud using the AHC method [36], which provides the plane coefficients (n,d) and supporting points in the point cloud for each plane instance. $n = (n_x, n_y, n_z)$ is the unit plane normal and d is the distance of the plane from origin. We further downsample the point cloud of each plane using voxel-grid filtering, with a voxel size of $0.2m$. Moreover, we discard potentially unstable plane observations, where the maximum point-plane distance between the plane and its points is larger than $0.04m$. For pose optimization, we use the minimal representation of planes: $q(\pi) = (\phi = arctan(\frac{n_y}{n_x}), \psi = arcsin(n_z), d)$, where ϕ and ψ are the azimuth and elevation angles of the plane normal. Plane matches are determined by comparing the angle between normals and the point-plane distance of planes.

C. Manhattan Frame Detection and Mapping

In contrast to using the popular mean-shift clustering algorithm [10], [11] for MF detection, which uses per-pixel normal estimation, we exploit the plane normals already extracted from the scene. An MF M_k can be represented by three mutually perpendicular plane normals (n^k_1, n^k_2, n^k_3). To detect an MF M_k in the current frame F_i, we check the angle between the plane normals n_z, where $n_z \in \{n_0 ... n_r\}$ is the normal of a detected plane and r is the total number of detected planes in F_i. An MF is detected whenever any three planes are mutually perpendicular. We can represent the observation of M_k in camera coordinates C_i of F_i with a rotation matrix

$$R_{c_i m_k} = \begin{bmatrix} n^k_1 & n^k_2 & n^k_3 \end{bmatrix}. \tag{1}$$

If only two perpendicular normals n^k_1 and n^k_2 are found, the third one n^k_3 can be recovered by taking the cross product between n^k_1 and n^k_2, thus the MF can be recovered from two planes as well.

Since sensor noise can lead to inconsistencies, where the columns of the matrix $R_{c_i m_k}$ are not orthogonal, we use SVD to approximate $R_{c_i m_k}$ with the closest rotation matrix $\hat{R}_{c_i m_k}$:

$$SVD(R_{c_i m_k}) = UDV^T, \tag{2}$$

$$\hat{R}_{c_i m_k} = UV^T. \tag{3}$$

Furthermore, we also build a Manhattan map G to collect MFs encountered in the scenes, where G stores both full and partial MF observations along with the corresponding frames in which they are observed:

$$G = \{M_k \mapsto F_i\}. \tag{4}$$

Building this map allows us to estimate drift-free rotation when we encounter MF M_k in any successive frame F_j.

To find a match between two observations of the same MF in camera frames F_i and F_j, we check for matches of their constituent planes to the map planes. Each map plane has a unique ID in the map. If planes of both observations are matched to the same map planes, determined by comparing IDs, then the observations belong to the same MF.

D. Pose Estimation

The camera pose ξ_{cw} consists of a rotation matrix $R_{cw} \in SO(3)$ and a translation vector $t_{cw} \in \mathbb{R}^3$, from world coordinates W to camera coordinates C. If MW assumption is not followed, determined by the failure to detect any MF, we estimate the full 6D pose by tracking features. In case of an MW scene, rotation and translation are decoupled and estimated separately.

*1) **For non-MW scenes**:* In non-MW scenes, points, lines and planes can be tracked to estimate a 6D camera pose. We define reprojection errors e_p, e_l and e_π between observed features and their corresponding matched 3D features in the map as

$$\begin{cases} e_p = p_{obs} - \Pi(R_{cw} P_w + t_{cw}) \\ e_l = {l_{obs}}^T \Pi(R_{cw} P^l_x + t_{cw}) \\ e_\pi = q(\pi_c) - q(T^{-T}_{cw} \pi_w) \end{cases}, \tag{5}$$

where Π is the projection function using the intrinsic camera matrix and P_x^l is an endpoint of the 3D line, with $x \in \{start, end\}$. We also find parallel and perpendicular plane matches for each observed plane, which are added as structural constraints $e_{\pi_\parallel}$ and $e_{\pi_\perp}$ to the overall energy function as

$$\begin{cases} e_{\pi_\parallel} &= q_n(n_c) - q_n(R_{cw} n_w) \\ e_{\pi_\perp} &= q_n(R_\perp n_c) - q_n(R_{cw} n_w) \end{cases}, \tag{6}$$

where n_c and n_w are the normals of the observed plane and matched plane landmark, $R_\perp$ is a 90° rotation matrix and $q_n(\pi) = (\phi, \psi)$.

Assuming a Gaussian noise model and combining all errors, the final energy function is written as $e = \sum \rho_y \left(e_y{}^T \Lambda_y e_y\right)$, where $y \in \{p, l, \pi, \pi_\parallel, \pi_\perp\}$ and Λ and ρ denote the inverse covariance matrix and the robust Huber cost function, respectively. This energy function is optimized using the Levenberg-Marquardt algorithm to get the optimal pose estimate $\xi_{cw}^* = \text{argmin}_{\xi_{cw}}(e)$.

*2) **For MW scenes**:* In structured MW scenes, we decouple pose estimation and use our novel approach to estimate drift-free rotation, while feature tracking is used for translation estimation. For rotation estimation, all MFs in the scene can be detected using the method described in Section III-C. For each detected MF M_k in frame F_i, represented by the corresponding rotation $R_{c_i m_k}$, we search for M_k in our Manhattan map G. If M_k is found in G, we can obtain the corresponding frame F_j from G, in which M_k was first observed. F_j serves as the reference frame, containing the MF observation $R_{c_j m_k}$ and pose estimate $\xi_{c_j w}$, which could have either been estimated by MF tracking or feature tracking.

Our goal here is to obtain the rotation $R_{c_i w}$ from world coordinates to current camera frame F_i. To achieve that, first, we use the observations of M_k in F_i and F_j to calculate the relative rotation between them as

$$R_{c_j c_i} = R_{c_j m_k} R_{c_i m_k}{}^T. \tag{7}$$

Then, we take the rotation estimate $R_{c_j w}$ from the pose estimate $\xi_{c_j w}$ of F_j and concatenate it with the relative rotation between F_i and F_j to get

$$R_{w c_i} = R_{c_j w}{}^T R_{c_j c_i}. \tag{8}$$

Finally, we transpose $R_{w c_i}$ to get our desired rotation $R_{c_i w}$.

Such rotation estimation is only possible if M_k has been already observed, i.e. M_k is stored in G. If F_i does not contain any previously observed MF, then we use the feature tracking method (Section III-D.1) to estimate the full pose. When an MF M_k is observed for the first time, we store it in our Manhattan map G, along with its frame F_i. For any subsequent observations of M_k, F_i can be used as a reference for rotation estimation. In case of multiple MF detections in F_i, we select the dominant MF, i.e. the one which contains the highest number of points in the point clouds of its constituent planes.

Now that we have the rotation R_{cw}, we want to find the corresponding translation t_{cw} that will give us the full camera pose ξ_{cw}. For this, we use feature tracking as described in Section III-D.1. The translation can be obtained by solving $t_{cw}^* = \text{argmin}_{t_{cw}}(e_t)$, where $e_t = \sum \rho_z \left(e_z{}^T \Lambda_z e_z\right)$ and $z \in \{p, l, \pi\}$. We fix rotation and only update the translation during the optimization process. Note that we do not use parallel and perpendicular constraints for planes here, since they only provide rotational information.

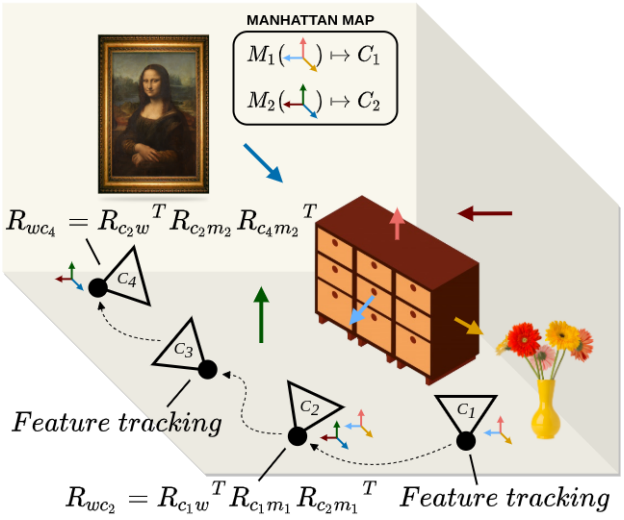

Fig. 3. A toy example showing two MFs and estimated pose for four camera frames. C_1 observes MF M_1, which is added to the Manhattan Map G. It uses feature tracking as there was no previous observation of M_1. C_2 observes both M_1 and M_2, so M_2 is added to G. Since M_1 is already present in G, a drift-free rotation is estimated for C_2 with C_1 as reference frame. C_3 does not observe any MF while C_4 observes M_2, which allows for a drift-free rotation using C_2 as reference frame.

E. Sparse mapping

Our SLAM system maintains a sparse map of landmarks and keyframes. For our sparse map, we follow the keyframe-based approach of [2], where a new frame is only added when it observes a significant number of new landmarks. New landmarks, i.e. points, lines and planes, are initialized and added to the map from keyframes using the depth map provided by the RGB-D image. If a new plane is matched to a previous map plane, we only update the point cloud of the map plane, otherwise, the new plane is added to the map. Following [2], we maintain a co-visibility graph among keyframes to determine the local map of the current frame and remove unreliable landmarks and redundant keyframes using respective culling strategies.

F. Dense surfel mapping

To improve the reconstruction efficiency, we provide a novel dense mapping strategy based on [14]. Instead of building a surfel for every pixel like ElasticFusion, [14] divides each image into superpixels and builds surfels based on the extracted superpixels. This approach reduces the number of surfels, allowing it to run on a CPU.

In our method, we further improve the efficiency of [14] by utilizing extracted planes in the scene. For planar regions, we

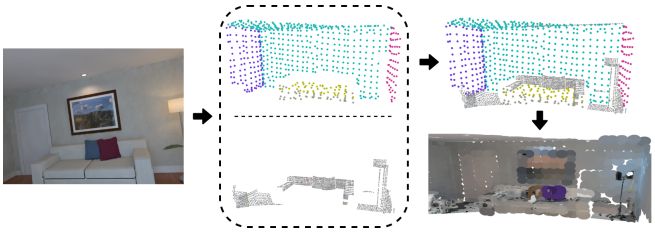

Fig. 4. **Left:** Sample scene. **Middle:** Points in sparse map planes and non-planar surfel points of the same scene. **Top Right:** Combined plane points and surfel points. **Bottom Right:** Rendered surfels.

build surfels by reusing planes from our sparse map, making our method more memory-efficient. We update the method provided by [14] as follows:

- Our plane detection method provides a mask for planar regions in the frame. We use this mask to generate superpixels for non-planar regions, using the modified SLIC [37] method of [14].
- Surfels are generated and fused for non-planar regions using the method of [14].
- For planar regions, we use the points from our sparse planes as surfel positions. Each surfel is assigned the normal of the corresponding plane. To determine radius of the surfel, we utilize the size of the voxel used to downsample our plane during voxel grid filtering. We take the length of the cross sectional diagonal of the voxel, divide it by two and set that as the radius.

IV. Evaluation

In this section, we evaluate multiple aspects of our system on publicly available datasets and compare it with feature-based methods ORB-SLAM2 and SP-SLAM, MW-based method L-SLAM and our previous MW-based works S-SLAM and RGBD-SLAM. All experiments are performed on an Intel Core i5-8250U CPU @ 1.60GHz $\times$ 8 with 19.5 GB RAM. We do not use any GPU for our experiments. Our method runs at around 15 Hz, taking 67 ms for tracking and 40 ms for superpixel extraction and surfel fusion (on a separate thread), on average. Additionally, we disable the bundle adjustment and loop closure modules of ORB-SLAM2 and SP-SLAM for a fair comparison.

A. Pose Estimation

1) ICL-NUIM: The ICL-NUIM [38] dataset provides camera sequences containing scenes for two synthetically-generated indoor environments: a living room and an office room. These environments contain large areas of low-texture surfaces like walls, ceiling, and floor. Table I shows the performance of our method based on translation ATE RMSE, compared to other feature- and MW-based SLAM systems. We also show the number of frames where MF tracking was used. Since ICL-NUIM is rendered based on a rigid Manhattan World model, MW-based methods work well, specially L-SLAM in of-kt0 and of-kt3 sequences and RGBD-SLAM [12] in lr-kt0 and of-kt2. However, MW-based methods are sensitive to the structure of environment as they need two perpendicular elements for every scene. In living room environments, especially in lr-kt3, some viewpoints are too close to the wall and contain only one plane, which leads to errors for MW-based approaches. Our method, however, is more robust as it switches to feature tracking in these cases, as well as in scenes where the detected planes are noisy. Feature-based methods ORB-SLAM and SP-SLAM also work well as both environments contain abundant texture. Nevertheless, our approach outperforms prior methods, by taking advantage of both structure and texture in the scene.

2) TUM RGB-D: The TUM RGB-D benchmark [39] is another popular dataset for the evaluation of SLAM algorithms. It consists of several real-world camera sequences which contain a variety of scenes, like cluttered areas and scenes containing varying degrees of structure and texture. MW-based systems struggle in cluttered environments, while point-based systems perform poorly in scenes lacking texture, so such a variety of scenes is suitable for showcasing how our system can robustly adapt to both MW and non-MW scenes.

In the fr1 and fr2 sequences where scenes are cluttered and contain few or no MFs, MW-based methods S-SLAM, RGBD-SLAM and L-SLAM cannot track as they need an MF for every frame, as shown in Table I. Instead, the proposed method can robustly estimate pose in non-MW scenes, performing equivalently to feature-based ORB-SLAM and SP-SLAM. For the fr3 sequence, our decoupled MW-based estimation gives improved results for structured environments. Four of the six tested sequences contain no or limited texture, resulting in a failure of ORB-SLAM2. SP-SLAM uses plane features as well, so it provides good results on all sequences except for 'cabinet'. On the other hand, MW-based S-SLAM and L-SLAM exploit structural information, although the lack of texture affects their translation estimation. RGBD-SLAM uses planes and structural constraints for translation estimation as well, so it works particularly well for 's-nt-far' and 'l-cabinet' sequences. As depth data in the TUM RGB-D sequences is captured from real-world scenes, it is not as accurate as sequences from the ICL-NUIM dataset. Hence, MW-based methods suffer due to noisy surface normals, especially in the cabinet and large-cabinet sequences. This affects our method as well, so to circumvent this, our method switches to feature tracking for frames with noisy planes.

B. Drift

To test the amount of accumulated drift and robustness over time, we evaluate our system on the TAMU RGB-D dataset [40] which contains long indoor sequences. Although the dataset does not provide ground-truth poses, the camera trajectory is a loop, so we can calculate the accumulated drift by taking the Euclidean distance between the starting and ending point of our estimated trajectory.

Table II shows the drift of our method, compared to ORB-SLAM2. Since TAMU RGB-D dataset has real-world scenes with noisy depth data, our method uses drift-free MF tracking only for frames with less noisy planes. We also evaluate

TABLE I

COMPARISON OF ATE RMSE (M) FOR ICL-NUIM AND TUM RGB-D SEQUENCES. × REPRESENTS TRACKING FAILURE. - MEANS RESULT IS NOT AVAILABLE.

		Methods						Frames	
Dataset	Sequence	Ours	S-SLAM [11]	RGBD-SLAM [12]	ORB-SLAM2 [6]	SP-SLAM [5]	L-SLAM [10]	Total	MF
ICL NUIM	lr-kt0	0.007	-	**0.006**	0.014	0.019	0.015	1510	1203
	lr-kt1	**0.011**	0.016	0.015	**0.011**	0.015	0.027	967	678
	lr-kt2	**0.015**	0.045	0.020	0.021	0.017	0.053	882	771
	lr-kt3	**0.011**	0.046	0.012	0.018	0.022	0.143	1242	1030
	of-kt0	0.025	-	0.041	0.049	0.031	**0.020**	1510	1505
	of-kt1	**0.013**	×	0.020	0.029	0.018	0.015	967	945
	of-kt2	0.015	0.031	**0.011**	0.030	0.027	0.026	882	786
	of-kt3	0.013	0.065	0.014	0.012	0.012	**0.011**	1242	1212
	Average	**0.014**	0.040	0.017	0.023	0.020	0.039		
TUM RGB-D	fr1/xyz	**0.010**	×	×	**0.010**	**0.010**	-	798	1
	fr1/desk	0.027	×	×	**0.022**	0.026	-	613	1
	fr2/xyz	**0.008**	×	×	0.009	0.009	-	3669	0
	fr2/desk	0.037	×	×	0.040	**0.025**	-	2965	26
	fr3/s-nt-far	0.040	0.281	**0.022**	×	0.031	0.141	793	688
	fr3/s-nt-near	**0.023**	0.065	0.025	×	0.024	0.066	1053	796
	fr3/s-t-near	0.012	0.014	-	0.011	**0.010**	0.156	1056	564
	fr3/s-t-far	0.022	0.014	-	**0.011**	0.016	0.212	906	576
	fr3/cabinet	**0.023**	-	0.035	×	×	0.291	1111	985
	fr3/l-cabinet	0.083	-	**0.071**	×	**0.074**	0.140	983	188

TABLE II

COMPARISON OF THE ACCUMULATED DRIFT (M) IN TAMU RGB-D SEQUENCES. -MF MEANS ONLY FEATURE TRACKING IS USED.

				Frames	
Sequences	Ours	Ours/-MF	ORB-SLAM2 [6]	Total	MF
Corridor-A	0.53	0.77	3.13	2658	401
Entry-Hall	0.39	0.81	2.22	2260	282

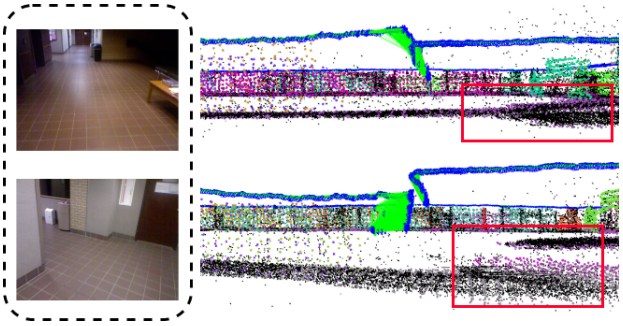

Fig. 5. Drift for TAMU-RGBD Corridor-A sequence. The pose estimates and reconstruction of our method around the loop point are shown. The red boxes highlight the inconsistency in ground plane when drift-free rotation is not used. **Left:** Sample scenes from the sequence. **Top Right:** Result with drift-free rotation enabled. **Bottom Right:** Result with drift-free rotation disabled.

the effect of our MF tracking method on the drift in pose estimates. Without MF tracking, our method still performs better than ORB-SLAM2, thanks to the addition of planes and structural constraints in the feature tracking module. With the addition of MF tracking proposed in our method, the drift of pose estimation is further reduced. It can be seen in Figure 5 that the reconstruction of floor aligns better at the loop point when MF tracking is enabled. These results indicate that drift could be further reduced with less noisy depth data, as it would result in more MFs being detected and used for drift-free rotation estimation.

TABLE III

RECONSTRUCTION ERROR (CM) ON THE ICL-NUIM DATASET.

Sequence	E-Fus [15]	InfiniTAM [41]	DSM [14]	Ours
lr-kt0	0.7	1.3	0.7	**0.5**
lr-kt1	0.7	1.1	0.9	**0.6**
lr-kt2	0.8	**0.1**	1.1	0.7
lr-kt3	2.8	2.8	1.0	**0.7**

C. Reconstruction Accuracy

Table III shows the reconstruction accuracy of our method evaluated on the living room sequences of the ICL-NUIM dataset. The evaluation is based on the point cloud generated by our surfels. ElasticFusion and InfiniTAM show good performance, with the latter getting an excellent result for lr-kt2. DSM [14], based on ORB-SLAM, performs admirably as the living room sequences have plenty of texture. Our method, however, uses structure in the environment and performs best on three out of four sequences. ElasticFusion and InfiniTAM need a GPU while DSM and our method work only on a CPU.

V. CONCLUSION

In this paper, we propose ManhattanSLAM, a method that tracks camera pose robustly in general indoor scenes, with the added ability of exploiting the structural regularities in MW scenes to compute low-drift pose estimates, as shown in our experiments. Furthermore, we exploit planar regions in the scene to provide an efficient surfel-based dense reconstruction of the environment. Future enhancements to the system can include adding a loop closure module, improving plane detection to further discard unstable observations and making the planar surfel radius flexible to more closely fit the actual plane boundary.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "Pl-slam: A stereo slam system through the combination of points and line segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[4] A. Zureiki and M. Devy, "Slam and data fusion from visual landmarks and 3d planes," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 14 651–14 656, 2008.

[5] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei, "Point-plane slam using supposed planes for indoor environments," *Sensors*, vol. 19, no. 17, p. 3795, 2019.

[6] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[7] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, "Real-time manhattan world rotation estimation in 3d," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1913–1920.

[8] Y. Zhou, L. Kneip, C. Rodriguez, and H. Li, "Divide and conquer: Efficient density-based tracking of 3d sensors in manhattan worlds," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 3–19.

[9] P. Kim, B. Coltin, and H. J. Kim, "Visual odometry with drift-free rotation estimation using indoor scene regularities." in *BMVC*, vol. 2, no. 6, 2017, p. 7.

[10] P. Kim, B. Coltin, and H. Jin Kim, "Linear rgb-d slam for planar environments," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 333–348.

[11] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-slam: Low-drift monocular slam in indoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[12] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari, "Rgb-d slam with structural regularities," 2020. [Online]. Available: https://arxiv.org/abs/2010.07997

[13] J. Straub, G. Rosman, O. Freifeld, J. J. Leonard, and J. W. Fisher III, "A mixture of manhattan frames: Beyond the manhattan world," in *CVPR*, 2014. [Online]. Available: http://people.csail.mit.edu/jstraub/download/straub2014mmf.pdf

[14] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense surfel mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6919–6925.

[15] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, pp. 1697 – 1716, 2016.

[16] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[17] G. Zhang and I. H. Suh, "Building a partial 3d line-based map using a monocular slam," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1497–1502.

[18] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "Structslam: Visual slam with building structure lines," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1364–1375, 2015.

[19] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo slam," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015.

[20] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 4503–4508.

[21] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5182–5189.

[22] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "Cpa-slam: Consistent plane-model alignment for direct rgb-d slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1285–1291.

[23] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

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

[25] P. Kim, B. Coltin, and H. J. Kim, "Low-drift visual odometry in structured environments by decoupling rotational and translational motion," in *2018 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2018, pp. 7247–7253.

[26] K. Joo, T.-H. Oh, F. Rameau, J.-C. Bazin, and I. S. Kweon, "Linear rgb-d slam for atlanta world," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1077–1083.

[27] 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 *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.

[28] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar slam," in *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE, 2014, pp. 157–164.

[29] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[30] S. Schreiberhuber, J. Prankl, T. Patten, and M. Vincze, "Scalablefusion: High-resolution mesh-based real-time 3d reconstruction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 140–146.

[31] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[33] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[34] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2008.

[35] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[36] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6218–6225.

[37] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[38] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2014, pp. 1524–1531.

[39] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

[40] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3934–3942.

[41] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray, "Infinitam v3: A framework for large-scale 3d reconstruction with loop closure," *arXiv preprint arXiv:1708.00783*, 2017.