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.

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.
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.

ii) 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 monoocular 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 II-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.
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 \((p^l_{\text{start}}, p^l_{\text{end}})\) and \((p'^l_{\text{start}}, p'^l_{\text{end}})\) respectively while also obtaining normalized line function for the observed 2D line as \(l_{\text{obs}} = p^l_{\text{start}} \times p'^l_{\text{end}} / \|p^l_{\text{start}}\| \|p'^l_{\text{end}}\| = (a, b, c)\). 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: \(q(p) = (\phi = \arctan(n_x / n_y), \psi = \arcsin(n_z), d)\), where \(\phi\) and \(\psi\) 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

\[
R_{c,m_k} = \begin{bmatrix} n^k_1 & n^k_2 & n^k_3 \end{bmatrix}.
\]

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 \(R_{c,m_k}\) are not orthogonal, we use SVD to approximate \(R_{c,m_k}\) with the closest rotation matrix \(\hat{R}_{c,m_k}\):

\[
\text{SVD}(R_{c,m_k}) = UDV^T,
\]

\[
\hat{R}_{c,m_k} = UV^T.
\]

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:

\[
G = \{M_k \mapsto F_i\}.
\]

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 \(\xi_{cw}\) 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 \(e_\pi\) between observed features and their corresponding matched 3D features in the map as

\[
\begin{align*}
\begin{cases}
    e_p = p_{\text{obs}} - \Pi(R_{cw}p_w + t_{cw}) \\
    e_l = l_{\text{obs}} - \Pi(R_{cw}l'_w + t_{cw}) \\
    e_\pi = q(p_c) - q(T_{cw}l'_w)
\end{cases}
\end{align*}
\]

where

\[
\begin{align*}
\begin{cases}
p_{\text{obs}} = p^l_{\text{start}} \times p'^l_{\text{end}} / \|p^l_{\text{start}}\| \|p'^l_{\text{end}}\| \\
l_{\text{obs}} = l_{\text{start}} \times l_{\text{end}} / \|l_{\text{start}}\| \|l_{\text{end}}\| \\
q(p_c) = q(p) = (\phi = \arctan(n_x / n_y), \psi = \arcsin(n_z), d)
\end{cases}
\end{align*}
\]
where $\Pi$ is the projection function using the intrinsic camera matrix and $P_e^i$ is an endpoint of the 3D line, with $x \in \{\text{start}, \text{end}\}$. We also find parallel and perpendicular plane matches for each observed plane, which are added as structural constraints $e_{\pi_i}$ and $e_{\pi_j}$ to the overall energy function as

$$
e_{\pi_i} = q_0(n_e) - q_0(R_{cw}n_w),$$
$$e_{\pi_j} = q_0(R_{e_j}n_e) - q_0(R_{cw}n_w),$$

(6)

where $n_e$ and $n_w$ are the normals of the observed plane and matched plane landmark, $R_{e_j}$ is a 90° rotation matrix and $q_0(\pi) = (\phi, \psi)$.

Assuming a Gaussian noise model and combining all errors, the final energy function is written as $e = \sum \rho_p(e_i^T \Lambda e_i)$, where $y \in \{p, l, \pi, \pi_l, \pi_p\}$ and $\Lambda$ and $\rho$ 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 $\tilde{\xi}_w = \text{argmin}_{\xi_w}(e)$.

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 $R_{e_j}$, we search for $M_k$ in our Manhattan map $G$. If $M_k$ is found in $G$, we 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 $R_{e_j}$ and pose estimate $\tilde{\xi}_j$, which could have either been estimated by MF tracking or feature tracking.

Our goal here is to obtain the rotation $R_{cw}$ 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

$$R_{e_j} = R_{e_j} R_{e_j}^T.$$ (7)

Then, we take the rotation estimate $R_{e_j}$ from the pose estimate $\tilde{\xi}_j$, and concatenate it with the relative rotation between $F_i$ and $F_j$ to get

$$R_{cw} = R_{e_j}^T R_{e_j}.$$ (8)

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

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 $\tilde{\xi}_{cw}$. For this, we use feature tracking as described in Section II-D.1. The translation can be obtained by solving $t_{cw} = \text{argmin}_{t_{cw}}(e_t)$, where $e_t = \sum \rho_p(e_t^T \Lambda e_t)$ 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.

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
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 × 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 1 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 algo-
rithms. 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 tex-
ture, 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 1. 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 trans-
lation 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 2 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.

| Dataset | Sequence | Ours | S-SLAM [11] | RGBD-SLAM [12] | ORB-SLAM2 [6] | SP-SLAM [5] | L-SLAM [10] | Frames |
|---------|----------|------|-------------|----------------|----------------|-------------|-------------|--------|
| ICL NUIM | lr-kt0 | 0.007 | 0.006 | 0.004 | 0.014 | 0.019 | 0.015 | 967 |
|         | lr-kt1 | 0.011 | 0.016 | 0.015 | 0.011 | 0.015 | 0.027 | 771 |
|         | lr-kt2 | 0.015 | 0.045 | 0.020 | 0.021 | 0.017 | 0.053 | 1242 |
|         | lr-kt3 | 0.011 | 0.046 | 0.012 | 0.018 | 0.022 | 0.143 | 1510 |
|         | of-kt0 | 0.025 | - | 0.041 | 0.049 | 0.031 | 0.020 | 1130 |
|         | of-kt1 | 0.013 | - | 0.020 | 0.029 | 0.018 | 0.015 | 967 |
|         | of-kt2 | 0.015 | 0.031 | 0.011 | 0.015 | 0.027 | 0.026 | 882 |
|         | of-kt3 | 0.013 | 0.065 | 0.014 | 0.012 | 0.012 | 0.011 | 1242 |
| Average | 0.014 | 0.040 | 0.017 | 0.023 | 0.020 | 0.039 | | |
| TUM RGB-D | fr2/xyz | 0.010 | - | - | 0.010 | 0.010 | - | 798 |
|         | fr1/desk | 0.027 | - | - | 0.022 | 0.026 | - | 613 |
|         | fr2/xyz | 0.008 | - | - | 0.009 | 0.009 | - | 3669 |
|         | fr2/desk | 0.037 | - | - | 0.040 | 0.025 | - | 2965 |
|         | fr3s-nt-far | 0.040 | 0.281 | 0.022 | - | - | - | 793 |
|         | fr3s-t-near | 0.023 | 0.065 | 0.025 | - | - | - | 1053 |
|         | fr3s-t-far | 0.012 | 0.014 | - | 0.011 | 0.010 | 0.156 | 1056 |
|         | fr3s-cabinet | 0.023 | - | 0.035 | - | - | 0.291 | 1111 |
|         | fr3M-cabinet | 0.083 | - | 0.071 | - | - | 0.074 | 983 |

TABLE II

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

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

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 |

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.

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 surfs. 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.
