SlamCraft: Dense Planar RGB Monocular SLAM

Jason Rambach, Paul Lesur, Alain Pagani, Didier Stricker
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
Jason.Rambach@dfki.de, Paul.Lesur@dfki.de, Alain.Pagani@dfki.de, Didier.Stricker@dfki.de

Abstract

Monocular Simultaneous Localization and Mapping (SLAM) approaches have progressed significantly over the last two decades. However, keypoint-based approaches only provide limited structural information in a 3D point cloud which does not fulfil the requirements of applications such as Augmented Reality (AR). SLAM systems that provide dense environment maps are either computationally intensive or require depth information from additional sensors. In this paper, we use a deep neural network that estimates planar regions from RGB input images and fuses its output iteratively with the point cloud map of a SLAM system to create an efficient monocular planar SLAM system. We present qualitative results of the created maps, as well as an evaluation of the tracking accuracy and runtime of our approach.

1 Introduction

Visual SLAM is a fundamental computer vision problem. Significant advances were made within the past 5 years and many successful SLAM systems were introduced each with its own strengths and weaknesses. Sparse keypoint based SLAM systems are sufficient for robotic navigation in unknown environments [11], however for AR and robotic interaction there are increased requirements in terms of environment mapping. In particular, AR applications can benefit from a dense 3D representation of the environment to allow interaction of the virtual objects with the real world and even semantic information to dictate the type of interaction with certain objects. Thus, a lot of research effort in the SLAM domain is shifting towards this direction [10].

To address this, dense mapping SLAM frameworks have been proposed, using either additional depth sensors or dense 3D reconstruction of keypoints which can be demanding in terms of computational effort with a real time constraint [12, 18]. The idea of using planar surfaces to represent the map is very attractive especially for indoor scenarios since it directly creates the ideal environment for the realistic placement of AR content. The idea was initially explored in [14] with an RGB-D system and in [3] for RGB.

In parallel, machine learning approaches utilizing convolutional neural networks (CNN) have shown impressive results on many computer vision problems such as semantic recognition and segmentation. Of particular interest for the SLAM problem are approaches that can estimate the structure of a scene from single RGB input images (e.g. [7]).

In this paper, we explore the potential of combining SLAM systems based on traditional projective geometry with the output of CNNs. We propose a novel fusion framework that is applied on a keypoint-based monocular SLAM system to combine its sparse point cloud with a CNN planar surface segmentation system in order to create an efficient and robust dense planar SLAM system suited for indoor AR without using any depth sensors. In detail our contributions in this paper are:

- An efficient dense planar SLAM framework using only monocular RGB input images.
- A compact surfel map representation of the map based on planarity information,
- Improved tracking accuracy through the direct point refinement by applying planarity constraints on keypoints.

2 Related Work

Keypoint-based SLAM systems such as ORB-SLAM [11] create sparse point clouds from multiple views of 2D image features while direct methods such as LSD-SLAM [5] minimize photometric error between images but are able to map only textured areas. DTAM [12] is a fully direct method that allows registration of a full 3D model of its map to the input images in real time through an efficient GPU implementation. Dense 3D mapping of the environment in SLAM is of great importance but comes at a large computational cost and is a challenging problem especially regarding the reconstruction of non-textured areas.

Higher level map representation can be a solution to this. In Slam++ [15] the map is created at object level using previously known 3D models. In [14], a map was represented by a set of bounded planes, using however depth information in an RGB-D system which simplifies the problem. Planar RGB SLAM systems were introduced in [2, 3], combining semi-dense SLAM mapping with 3D superpixels of low-gradient regions achieving high quality mapping on a limited selection of sequences without a tracking accuracy evaluation. Similarly, [13] presents a comprehensive SLAM framework with tracking based on plane correspondences

which is however not real-time capable. In contrast, our approach aims at an efficient and accurate mapping of planar regions instead of a costly fully dense reconstruction and is able to track in planar as well as non-planar areas.

Within the recent advance of convolutional neural network (CNN) learning methods, many approaches that attempt to recover structural information of the scene from single images have been proposed. Among those, scene segmentation approaches are very popular [1] together with depth estimation approaches from RGB images [7]. The latter ones could potentially replace depth sensors, however their ability to be successfully trained across different datasets and cameras has not yet been fully verified. Of significant interest is the room layout estimation (i.e. segmentation of room walls) of RoomNet [8] and the very recent planar area segmentation of PlaneNet [9].

Even though the potential is obvious, there are not many existing approaches attempting to improve SLAM systems by combining them with CNN scene estimation. In CNN SLAM [17], predicted depth images from a CNN are used to improve the mapping of a direct monocular SLAM system. Practically, [17] is simulating an RGB-D system to achieve an impressive mapping quality, while in this work we are targeting at lightweight map representation in terms of the main planar areas which is of interest for applications such as AR. The importance of semantic labelling and object level segmentation in SLAM systems is addressed in [10].

3 Proposed approach

3.1 Notation

We define a 3D point $\boldsymbol{m}$ in a coordinate system A as $\boldsymbol{m}_A \in \mathbb{R}^3$. A plane $\boldsymbol{\mathcal{P}}_i$ is defined by its equation vector $\boldsymbol{p}_i = (a, b, c, d)$ so that for every point $\boldsymbol{m} = (x, y, z)$ on the plane the equation $ax + by + cz + d = 0$ holds true. A normal vector on the plane is then $\boldsymbol{n}_i = (a, b, c)$. Each plane $\boldsymbol{\mathcal{P}}_i$ holds a set of associated 3D points in the world coordinate system W, $\boldsymbol{\mathcal{P}}_{(M,i)} = \{\boldsymbol{m}_{(W,1)}, \boldsymbol{m}_{(W,2)}, \cdots, \boldsymbol{m}_{(W,N)}\}$, and a set of surfels $\boldsymbol{\mathcal{P}}_{S,i} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_K\}$. For each surfel, its 3D position is required, which is given as a quantized displacement on the plane surface originating from a 3D position on the plane which is common for all surfels.

The distance of a point $\boldsymbol{m}$ to a plane $\boldsymbol{\mathcal{P}}_i$ is given by $|l_{(\boldsymbol{\mathcal{P}},\boldsymbol{m})}|$ where $l_{(\boldsymbol{\mathcal{P}},\boldsymbol{m})} = \frac{\boldsymbol{n}^\top \boldsymbol{m} + d}{\sqrt{a^2+b^2+c^2}}$. The projection of the point on the plane is given by $\hat{\boldsymbol{m}} = \boldsymbol{m} - l_{(\boldsymbol{\mathcal{P}},\boldsymbol{m})} \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}$.

3.2 System Architecture

In Figure 1 we present the outline of SlamCraft, our dense planar SLAM approach. The keypoint-based ORB-SLAM [11] runs on every input image. Whenever

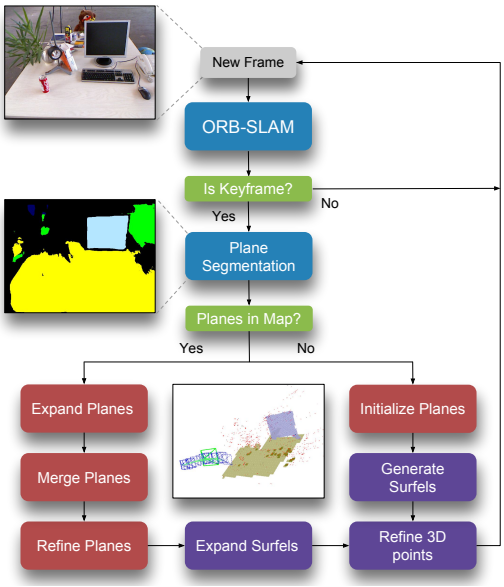

Figure 1. Architecture of the proposed Planar SLAM system

the SLAM system records a keyframe, a new thread is created with the task of executing the modules of our approach. We use a separate thread so that the frame-to-frame tracking of the SLAM system is not obstructed. Additionally, we perform segmentation only on every keyframe since applying our method on every frame would not be feasible in real-time mainly because of the processing time of the forward phase of the neural network.

On every keyframe image, a planar segmentation CNN [9] is applied. The output of that is a 2D segmentation of the image into different planar areas and non-planar regions. In the next step, we use the segmented image to assign the 3D points of the SLAM point-cloud map to planes (red modules in Figure 1).

These steps create plane equations and assign 3D points to the planes, however there is still no information about the planes boundaries that can be used for a dense 3D mapping. This is done in the subsequent steps (purple modules in Figure 1) that manage the surfel map.

3.3 Plane Segmentation

A CNN is trained to output a segmentation of planes from an RGB input image using the PlaneNet implementation [9]. We use a model that was trained on 50,000 annotated images of the Scannet dataset [4]. On Figure 2 we show examples of the output of the network on the images from the TUM RGB-D dataset [16]. The network is successful at extracting the dominant planes from the images such as the desk. For other planes however such as the computer monitor in the first row of images, some inconsistencies between frames can be

seen. As seen on the second row the network has some difficulties with the objects on the desk as they are sometimes included in the plane and other times not. In terms of mapping the output of the network although very informative cannot be used directly for 3D mapping and that is the reason why a comprehensive fusion scheme with the point cloud of the SLAM system is necessary to deal with temporal inconsistencies. PlaneNet [9] also provides plane parameters however these were not used as their accuracy could not be verified.

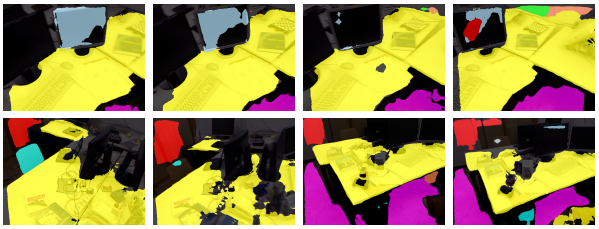

Figure 2. Planar segmentation results on images from the TUM RGB-D dataset

3.4 Plane Estimation

A map of existing planes denoted as $\mathcal{M} = \{\boldsymbol{\mathcal{P}}_1, \boldsymbol{\mathcal{P}}_2, \cdots \boldsymbol{\mathcal{P}}_N\}$ is created and maintained by our the system. The map $\mathcal{M}$ is initialized using the first keyframe segmentation and the 3D points that are visible on the frame together with their 2D position in the frame. For each 3D point an area of a few pixels (5×5) around its 2D image position in the segmented image is selected and if the segmentation label of all pixels in the area is consistent then this 3D point is added as a candidate point for a new plane. Thus, for every plane appearing in the segmented image we obtain a list of candidate 3D points. Using this list we perform plane fitting using Singular Value Decomposition (SVD) within an iterative RANSAC framework for outlier removal aiming to find a set of points to minimize the average distance to the estimated plain.

Existing planes have to be maintained and new planes have to be added using newly selected keyframes. Therefore we first try to match the keypoints observed in the frame to existing planes. If a keypoint does not have an associated plane, we examine the distance of the point to all the planes and in case the distance to one of the planes is below a threshold, we add the keypoint to that plane. For all the remaining keypoints, we follow the same procedure as in the initialization, where we match labels and keypoints in order to create plane candidates and possibly add new planes.

It is possible that parts of a larger plane are generated completely independently as two different planes (e.g. when a plane is visible in two keyframes with no common keypoints). In order to deal with these cases we evaluate every pair of planes to decide if they should be merged. For two planes $\boldsymbol{\mathcal{P}}_i, \boldsymbol{\mathcal{P}}_j$ to be merged two conditions need to apply. First, the planes should be almost parallel which holds when $|cos(\theta)| = |\frac{\boldsymbol{n}_i \boldsymbol{n}_j}{\|\boldsymbol{n}_i\| \|\boldsymbol{n}_j\|}| = 1$, where θ is the angle between the planes. Therefore, we require $|cos(\theta)| < \text{angle}_{\text{thr}}$ where $\text{angle}_{\text{thr}}$ is set to 0.95 in our implementation. If the planes are found parallel, the second condition addresses the distance between them which is also thresholded. After merging two planes the plane equation is re-evaluated using all its points.

It is possible that 3D points are added to a plane but also that 3D points are erased from the map by the SLAM system (i.e. by the culling modules of ORB-SLAM). Therefore the plane equation is refined using the current set of points. To achieve this we use SVD plane fitting on the entire current set of points in the plane. If the average distance error of points to the newly estimated plane equation is below a threshold then the plane equation is updated, otherwise the plane is set as invalid and is removed from the map of planes.

The 3D points of the SLAM system are often inaccurate after triangulation especially before bundle adjustment is performed. However, using planarity information from our system we can refine their 3D positions. To do this we replace the 3D position of a point $\boldsymbol{m}$ with its projection $\hat{\boldsymbol{m}}$ to its belonging plane $\boldsymbol{\mathcal{P}}$, given by $\hat{\boldsymbol{m}} = \boldsymbol{m} - l_{(\boldsymbol{\mathcal{P}}, \boldsymbol{m})} \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}$. By performing this refinement on some of the point cloud points (i.e. the ones that belong to a plane), we can correct points before the bundle adjustment and thus increase the pose estimation accuracy of the tracking module. This claim is also supported by results in the experiments Section 4.2.

3.5 Probabilistic Surfel Map

Maintaining a list of 3D points $\boldsymbol{\mathcal{P}}_{(M,i)}$ that belong to a plane $\boldsymbol{\mathcal{P}}_i$ does not give any concrete information on the size and boundaries of the plane in 3D space by itself. For this reason we additionally maintain a set of surfels $\boldsymbol{\mathcal{P}}_{(S,i)}$ for every plane. A surfel consists of a 3D point which indicates that a square area centered on that point belongs to the plane. In order to be able to dynamically update and remove surfels, each surfel $\boldsymbol{s}_i$ also has a probability of existence ρ_i.

Whenever a new plane is created, a centroid $\boldsymbol{c}$ and two perpendicular base vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ parallel to the plane are defined. First, surfels are added at the positions of the plane registered 3D points. For fast access, the surfels are indexed with a key defined by their displacement α, β along $\boldsymbol{v}_1, \boldsymbol{v}_2$ respectively so that the surfel position is given by $\boldsymbol{s}_i = \boldsymbol{c} + \alpha \boldsymbol{v}_1 + \beta \boldsymbol{v}_2$. This key is created by concatenating the two (integer) components $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into one 64 bits key, k.

Starting from these surfels an algorithm that traverses the 3D space on the plane is initiated. The algorithm expands the surfels by moving incrementally along the base vectors, projecting the new 3D position

onto the image and examining the label in the segmented image in order to add a surfel. This procedure allows mapping planes even in non-textured regions as long as there are a few starting keypoints on the plane. The probablity of new surfels is set to $\rho_i = 0.7$.

For already existing planes, first new surfels are created for any newly added 3D keypoints in the plane. Subsequently, all existing surfels 3D points are projected to the current keyframe. The label with most appearances is set as the correct plane label. Using this, the surfels probabilities are updated as follows: If the surfel has the correct label in the current frame then its probability is increased as $\bar{\rho}_i = \lambda\rho_i + (1 - \lambda)$, otherwise it is decreased as $\bar{\rho}_i = \lambda\rho_i$. λ is a forgetting factor that we set to 0.8 in our implementation. If the probability of a surfel drops below 0.5, then that surfel is deleted. The map of surfels is then expanded using the aforementioned map traversal procedure.

4 Experimental Results

In this section we present qualitative mapping and quantitative tracking accuracy results of our proposed system in the TUM RGB-D [16] and ICL-NUIM [6] datasets. No depth images from the datasets were used in our approach and the neural network we used for plane segmentation (Section 3.3) was not trained on images from these datasets.

4.1 Mapping evaluation

In Figure 3 we give some exemplary screenshots of the execution of SlamCraft on sequences from the TUM RGB-D dataset. We show selected frames with the current planar surfel map projected on them as well as the map on its own. Different colours are used for each plane. In the first and second row (sequence *fr2_xyz*) the first plane that is created is the PC monitor and shorty thereafter the desk is added to the plane map. The plane borders are correctly defined thanks to the probabilistic surfel map even though the network segmentation output shows some instability and temporal inconsistency (see Figure 2). This is also the reason why some surfels of the monitor are lost in later frames. As the camera moves the desk surface is correctly expanded, even with parts of the desk that are initially occluded by the monitor. The flatter objects on the table, such as the keyboard are included in the plane while some taller objects such as the headphones or the phone are excluded. Sequence *fr1_desk* in rows 3,4 covers a larger area and contains faster movement. The monitors were not registered here as planes due to the absence of sufficient map points on them to initialize a plane. The desk structure of the room was mapped successfully while excluding non-planar areas.

Results from two more sequences are given in Figure 4. The first row (TUM RGB-D sequence *fr3 structure texture far*), can be used for comparison to the work of [3]. We can see that the structure in the dataset is mapped correctly in all cases with similar quality to [3] and at a significantly lower computational cost (≈ 250 vs. ≈ 450 ms). In the second (ICL-NUIM sequence *lr kt0*) the two main planes (i.e. two walls of the room) are correctly mapped by our approach.

4.2 Tracking evaluation

In Table 1 we compare the tracking accuracy of the original ORB-SLAM to our SlamCraft. We present the mean and median of the RMSE error out of 20 runs of the two SLAM algorithms on 6 sequences from the TUM RGB-D dataset. We also give the relative difference (improvement or deterioration) of SlamCraft with respect to ORB-SLAM in %. The results verify that the point refinement using the planar information is of significant benefit to the tracking accuracy. For 5 out of 6 sequences SlamCraft has improved accuracy which is often more than 10% compared to the original ORB-SLAM. This is a welcome by-product of the planar mapping in our system, provided mainly by the refinement of points using planar constraints.

A runtime analysis (see Table 2) showed very similar results to ORB-SLAM which indicates that the system is real-time capable. All newly introduced mapping modules have very low runtime, apart from the neural network that requires about 250ms for the forward phase meaning that about 4 keyframes per second can be processed. This is sufficient since as we observed the SLAM system creates keyframes of about 10% of the total frames.

5 Conclusion

In this paper we presented a novel approach for a monocular RGB Planar SLAM system. The sparse point cloud of a keypoint-based SLAM System is fused with the output of a CNN that performs segmentation of planar areas from single RGB images. The proposed framework, consisting of the assignment of keypoints to planes and coverage of planar areas using a probabilistic surfel map, allows for an efficient creation of a dense planar map of the environment which is of great importance for applications such as AR. Furthermore, the refinement of points using the planar constraints is shown to significantly improve the SLAM localization accuracy compared to ORB-SLAM. Possible further work could include the mapping of non-planar areas or the improvement and speed-up of the segmentation output of the neural network.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on*

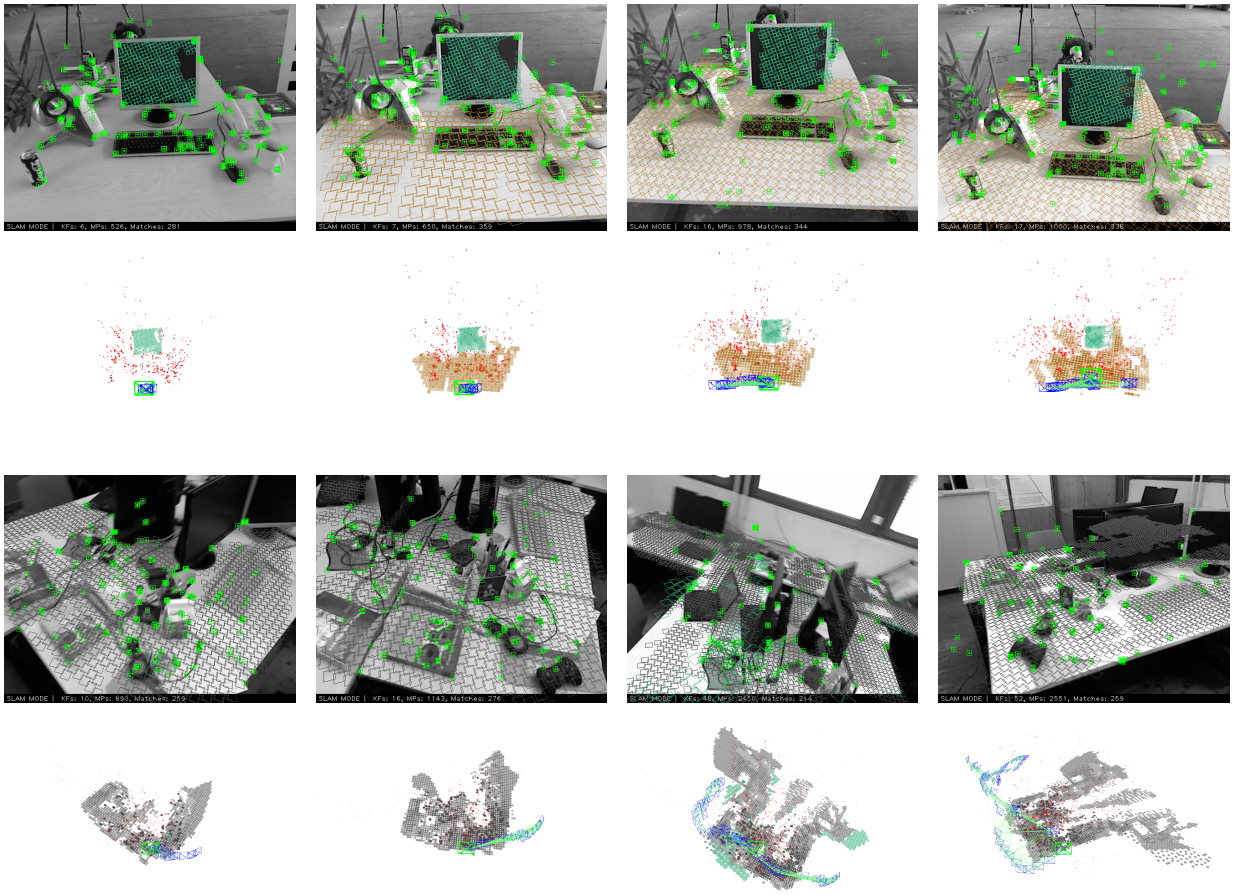

Figure 3. SlamCraft mapping on TUM RGB-D dataset (*fr2_xyz* and *fr1_desk*). Live frames with reprojected map and 3D map visualization

Table 1. Comparison of ORB-SLAM to SlamCraft tracking RMSE error on the TUM RGB-D dataset

Absolute KeyFrame Trajectory RMSE (cm)						
Seq.	**ORB-SLAM**		**SlamCraft**		**Difference %**	
#	Mean	Med.	Mean	Med.	Mean	Med.
fr1 xyz	0.76	0.62	**0.66**	**0.61**	-13.2	-1.6
fr2 xyz	0.13	0.14	**0.12**	**0.12**	-7.7	-14.3
fr1 desk	0.68	0.68	**0.60**	**0.47**	-11.8	-30.9
fr2 desk	**0.88**	**0.90**	0.89	0.91	+1.1	+1.1
fr3 sit_xyz	0.32	0.32	**0.31**	**0.31**	-3.1	-3.1
fr3 str_tex_far	0.86	0.87	**0.84**	**0.84**	-2.3%	-3.4

pattern analysis and machine intelligence, 39(12):2481–2495, 2017.

[2] A. Concha and J. Civera. Using superpixels in monocular SLAM. In *IEEE International Conference on Robotics and Automation (ICRA),*, pages 365–372. IEEE, 2014.

[3] A. Concha and J. Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotic Systems.*, number ART-2015-92153, 2015.

[4] A. Dai, A. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017.

[5] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[6] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE international conference on Robotics and automation (ICRA),*, pages 1524–1531. IEEE, 2014.

[7] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari,

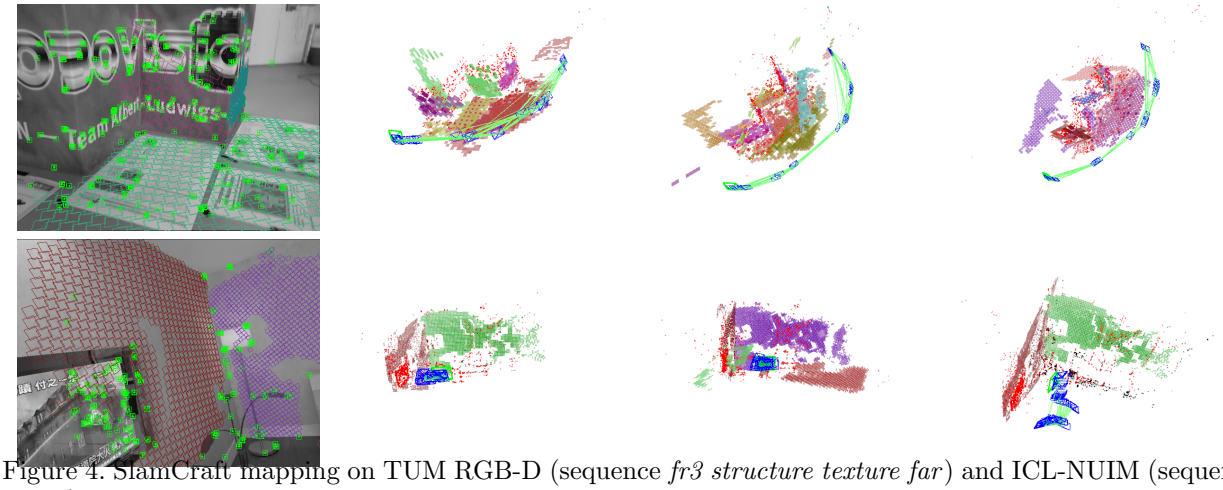

Figure 4. SlamCraft mapping on TUM RGB-D (sequence *fr3 structure texture far*) and ICL-NUIM (sequence *lr kt0*)

Table 2. Runtime comparison of original ORB-SLAM to SlamCraft on the TUM RGB-D dataset on a Desktop PC with an Intel(R) Xeon(R) 4 Cores CPU @ 3.70GHz with 32GB RAM and an NVidia GeForce GTX 1050 Ti GPU

Runtime comparison of ORB-SLAM modules (ms)		
Module	**ORB-SLAM** Mean	**SlamCraft** Mean
Local Mapping	194.72	202.402
Tracking	8.959	10.523

SlamCraft modules runtime	
Module	**time (ms)**
Plane Segmentation	254.38
ExpandPlanes	1.997
MergePlanes	0.040
RefinePlanes	0.024
ExpandSurfels	1.302
RefineMapPoints	0.074
Total	261.409

and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *Fourth International Conference on 3D Vision (3DV),*, pages 239–248. IEEE, 2016.

[8] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. RoomNet: End-To-End Room Layout Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4865–4874, 2017.

[9] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. PlaneNet: Piece-wise Planar Reconstruction from a Single RGB Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[10] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2017.

[11] R. Mur-Artal, J. M. M. Montiel, and J. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[12] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV),*, pages 2320–2327. IEEE, 2011.

[13] C. Raposo and J. Barreto. π Match: Monocular vSLAM and Piecewise Planar Reconstruction Using Fast Plane Correspondences. In *European Conference on Computer Vision*, pages 380–395. Springer, 2016.

[14] R. Salas-Moreno, B. Glocken, P. Kelly, and A. Davison. Dense planar SLAM. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164. IEEE, 2014.

[15] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359. IEEE, 2013.

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[17] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574. IEEE, 2017.

[18] T. Whelan, R. Salas-Moreno, B. Glocker, A. Davison, and S. Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.