Real-time Monocular Dense Mapping for Augmented Reality

Tangli Xue[†], Hongcheng Luo[†], Danpeng Cheng[§], Zikang Yuan[†], Xin Yang [†*]

[†] Huazhong University of Science and Technology, Wuhan, China
[§] University of Bridgeport, USA
[†]{joytl1993, hongcheng, U201414516, xinyang2014}@hust.edu.cn
[§]cdp0052@qq.com

ABSTRACT

Monocular simultaneous localization and mapping (SLAM) is a key enabling technique for many augmented reality (AR) applications. However, conventional methods for monocular SLAM can obtain only sparse or semi-dense maps in highly-textured image areas. Poorly-textured regions which widely exist in indoor and manmade urban environments can be hardly reconstructed, impeding interactions between virtual objects and real scenes in AR apps. In this paper, we present a novel method for real-time monocular dense mapping based on the piecewise planarity assumption for poorly textured regions. Specifically, a semi-dense map for highly-textured regions is first calculated by pixel matching and triangulation [6, 7]. Large textureless regions extracted by Maximally Stable Color Regions (MSCR) [11], which is a homogeneous-color region detector, are approximated using piecewise planar models which are estimated by the corresponding semi-dense 3D points and the proposed multi-plane segmentation algorithm. Plane models associated with the same 3D area across multiple overlapping views are linked and fused to ensure a consistent and accurate 3D reconstruction. Experimental results on two public datasets [15, 23] demonstrate that our method is 2.3X∼2.9X faster than the state-of-the-art method DPPTAM [2], and meanwhile achieves better reconstruction accuracy and completeness. We also apply our method to a real AR application and live experiments with a hand-held camera demonstrate the effectiveness and efficiency of our method in practical scenario.[1]

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics;

KEYWORDS

monocular dense mapping, plane model, augmented reality, multi-plane segmentation

*corresponding author Xin Yang, email: xinyang2014@hust.edu.cn
[1]A demo video is provided in the supplementary material

MM '17, , October 23–27, 2017, Mountain View, CA, USA.

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.1145/3123266.3123348

1 INTRODUCTION

Acquiring the relative camera pose between a moving camera and a scene and meanwhile estimating 3D structure of the scene, i.e. also named Simultaneous Localization and Mapping (SLAM) in the area of robotics, are two key tasks for applying AR applications in an unknown environment. The monocular camera stands out as one of the most convenient sensors for SLAM due to its simplicity to use, superior size, weight and power characteristics, low cost and large deployment in personal electronic devices (e.g. smartphones, tablets, small drones, etc.). A SLAM system relies solely on a monocular camera is also denoted as monocular SLAM.

During the past decades, several efforts have been made for accurate, robust, efficient monocular SLAM. One of the most notable works is PTAM [17] which proposed to split the two tasks of SLAM, i.e. tracking and mapping, into two separate parallel threads, and updated maps for only keyframes. The implementation style of parallelizing tracking and mapping and the utilization of keyframe-based map management enabled PTAM to achieve real-time speed with a satisfactory accuracy and robustness on a standard dual-core CPU, and these two strategies are widely used by most of modern visual SLAM systems such as ORB-SLAM [20] and LSD-SLAM [6], and visual odometry systems such as SVO [12] and DSO [5]. Despite advances in monocular SLAM, most existing systems can only provide a sparse or semi-dense map of a scene, in which depths of only points with sufficiently large gradients are obtained. A sparse or semi-dense map is sufficient for localization while not convenient for AR applications in which virtual objects usually need to interact with the physical scene. However, estimating a fully dense map is quite challenging for conventional monocular SLAM especially for textureless regions, such as surface of a table, black computer screen, etc., which commonly exist in indoor and urban manmade environments. This is because in most conventional SLAM systems, maps are constructed by finding correspondences between frames and then applying triangulation to correspondences for depth estimation. Thus, the computational complexity increases linearly with the density of a map, yielding difficulties in achieving both efficiency and a high map density. Additionally, the great difficulties in finding reliable correspondences in regions with little texture limit the amount of pixels for accurate depth estimation.

In order to achieve dense mapping, DTAM [21] performs pixel-wise depth estimation by minimizing an energy function composed of a data term which computes photometric errors between corresponding pixels and a regularization term which enforces smoothness of estimated depth. The expensive optimization procedure is accelerated by powerful GPUs for real-time performance.

However, DTAM requires rich texture and large-parallax camera motions for accurate depth estimation, it still performs poorly at large textureless regions. In recent years, some efforts have been made to integrate semantic information based on object recognition [3, 18, 22] or structure/layout of a scene [10] with monocular SLAM for dense mapping. That is once a particular object is recognized the prior shape information of the object is employed to densify the scene map in SLAM. However, accurate object recognition is also a challenging problem; incorrect recognition results could degrade the performance of depth estimation. Several recent works proposed handling high-gradient and low-gradient regions separately to achieve both a good reconstruction accuracy and map density in real-time. For instance, in [14] the authors proposed a multi-level mapping which partitions an image adaptively according to the gradient magnitudes in a region: high-gradient regions are represented at higher resolutions to capture fine details, while low-texture regions are represented at coarser resolutions to approximate planar structure. The most similar to us is [2] in which a dense mapping system named DPPTAM is proposed by exploiting planar structure. However, the performance of DPPTAM highly relies on extraction of superpixels [1] which is usually time consuming and has a low repeatability among frames. In addition, it is hard to guarantee that each superpixel consists of a single planar object. As a result, several reconstructed superpixels could be discarded due to the existence of multiple planar/non-planar regions, low repeatability of superpixels on other frames, yielding nontrivial and unnecessary cost. In the domain of computer vision, several methods [4, 13, 19] have been proposed to perform dense mapping from a single image based on convolutional neural networks (CNNs). However, the accuracy of depth estimation using CNNs is much lower than motion stereo. Additionally, CNN-based methods are computationally expensive and require powerful GPUs for acceleration. How to integrate CNN-based method into conventional SLAM systems is also a very challenging problem.

In this paper, we present a new system which is inspired by DPPTAM yet with three important modifications to enable an up to 2.9X speedup, better map completeness and accuracy comparing to DPPTAM: 1) We replaced the superpixel method of DPPTAM with a much more efficient and repeatable homogeneous region detector, called MSCR [11], for extracting candidate planar regions (CPRs). An ultrafast method based on dual projection of 3D semi-dense points is applied for matching CPRs among keyframes; only CPRs that can be reliably matched across consecutive keyframes are used for depth estimation, reducing unnecessary cost for constructing 3D structure for non-reliable CPRs. 2) We performed multi-plane segmentation to reconstruct all piecewise planar regions in each CPR. Iterative RANSAC [9] is applied to sparse 3D points located in each CPR for multi-plane model estimation. Pixels which lie in the CPR while do not have 3D depth information are assigned to one of the estimated plane models or removed as outliers. In contrast, in DPPTAM the superpixels containing multiple planes and non-planar objects are discarded, yielding a relative low utilization of superpixels. 3) Piecewise planar regions which are visible across multiple overlapping keyframes are linked and fused for a more accurate and consistent reconstruction. Experimental results on two public datasets, i.e. TUM [23] and [15] demonstrate that our dense mapping method outperforms the state-of-the-art method DPPTAM [5] in terms of faster speed and, better map completeness and accuracy. We also applied our method to an AR application to demonstrate its effectiveness in practical AR scenarios.

2 PRELIMINARY

2.1 Notation

We define $\boldsymbol{I}_t : \Omega \rightarrow \mathbb{R}$ image captured at time t, where $\Omega \subset \mathbb{R}^2$ represents the image pixel domain. We denote the camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3x3}$, the camera pose at frame t with respect to the world coordinates as:

$$\mathbf{T}_w^t = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{1}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$. The coordinate of a 3D point in the world coordinates is denoted as $\mathbf{P}^w = (x, y, z)^T \in \mathbb{R}^3$ and its projection to a 2D frame i is 2D coordinates in image coordinates denoted as $\mathbf{u} = (u, v)^T \in \Omega$ which can be computed based on the camera projection model $\boldsymbol{\pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\mathbf{u} = \boldsymbol{\pi}(\mathbf{T}_w^t \mathbf{P}^w) \tag{2}$$

The projection model $\boldsymbol{\pi}$ is determined by the intrinsic camera parameters $\mathbf{K}$. Similarly, the 3D points $\mathbf{P}^c$ in camera coordinates can be recovered from their 2D projections $\mathbf{u}$ by the inverse projection model $\boldsymbol{\pi}^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\mathbf{P}^c = \boldsymbol{\pi}^{-1}(\mathbf{u}, \rho_u) \tag{3}$$

where $\rho_u \in \mathbb{R}$ represents the depth of 2D point $\mathbf{u}$ in the camera frame.

2.2 Homography Model

Denote a 3D plane model $\Pi = \left[\mathbf{n}^T, d\right]$, where $\mathbf{n}$ is the unit normal vector of the plane Π and d is the distance along the normal of plane Π to the origin in the current camera. Every pixel $\mathbf{u}^k$ inside a plane Π in frame k can be mapped to frame j via the homography matrix $\mathbf{H}_\Pi$ in homogeneous coordinates as:

$$\mathbf{u}^j = \mathbf{H}_\Pi \mathbf{u}^k \tag{4}$$

where $\mathbf{H}_\Pi$ is induced by plane model Π:

$$\mathbf{H}_\Pi = \mathbf{K}(\mathbf{R} + \mathbf{t}\mathbf{n}_\Pi / d_\Pi)\mathbf{K}^{-1} \tag{5}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the relative rotation and translation between frame k and frame j, $\mathbf{K}$ is the camera intrinsic matrix.

3 METHOD

Figure 1 shows the overview of our system, which runs three parallel threads: tracking, semi-dense mapping and dense mapping. The tracking thread tracks the camera pose for every frame and selects keyframes for the subsequent semi-dense and dense mapping (Sec. 3.1). The semi-dense mapping thread estimates depth of high-gradient pixels which are then used for dense mapping of piecewise planar regions (Sec. 3.2). The dense mapping thread takes the semi-dense map, keyframes and camera poses of every keyframe as inputs and outputs a dense map based on the piecewise planarity assumption (Sec. 3.3). Specifically, the dense mapping

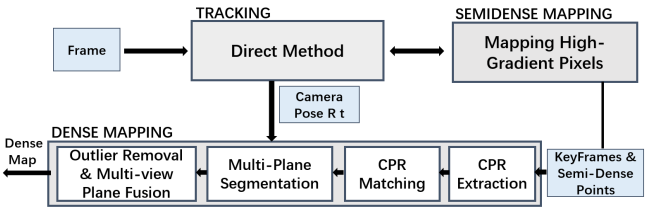

Figure 1: Overview of our system which consists of three main modules running on three parallel threads: tracking, semi-dense mapping for high-gradient pixels and dense mapping for textureless regions.

module consists of four main components: First, candidate planar regions (CPRs) are extracted at every keyframe based on a blob detector named maximally stable color regions (MSCR). Second, CPRs are matched across different keyframes and for those which can be repeatability detected and reliably matched in consecutive keyframes are identified for further processing. Third, we perform multi-plane segmentation for each CPR based on the semi-dense map and then assign the remaining pixels within the CPR to one of the reconstructed planes or non-planar regions to form a dense map. Finally, plane models which are visible across different overlapping views are linked for outlier removal and multi-plane fusion, yielding a more accurate and consistent 3D reconstruction. In the following, we detail each step.

3.1 Direct Method for Pose Tracking

In our method we employ a direct method [6, 12] for pose tracking due to its low computational complexity and high robustness. Specifically, the direct method estimates the camera pose by minimizing photometric errors of high-gradient pixels at the corresponding reprojection locations:

$$\mathbf{T}_k^n = \arg\min_T \sum_\Omega w_i \boldsymbol{\delta}_i^2 \tag{6}$$

where $\mathbf{T}_k^n$ represents the transformation from last keyframe k to current frame n, $\boldsymbol{\delta}^2$ is the intensity residual defined as the photometric difference between pixel i in frame n and its reprojection location in last keyframe k:

$$\boldsymbol{\delta}_i = I_n(\boldsymbol{\pi}(\mathbf{T}_k^n \cdot \boldsymbol{\pi}^{-1}(\mathbf{u}_i, \rho_i))) - I_k(\mathbf{u}_i) \tag{7}$$

To increase robustness to occlusion and to remove outliers, each residual is weighted with w_i as proposed in [16]. For each high-gradient pixel $\mathbf{u}_i$ with known depth ρ_i in keyframe k, we back-project the pixel to 3D coordinates and then reproject it into frame n. Eq. (6) can be interpreted a nonlinear weighted least-squares problem which can be solved by a standard Gauss-Newton algorithm.

3.2 Semi-dense Mapping for High-Gradient Pixels

The semi-dense mapping thread estimates the high-gradient pixel depth through pixel matching and the triangulation method. Mapping accuracy mainly depends on the accuracy of pixel matching. As stated in previous studies [7], a small baseline gives a unique but imprecise matching result and thus we select frames whose baseline between the last keyframe is over a threshold for depth estimation. For each high-gradient pixel with unknown depth in a keyframe, we exhaustively search its matching pixel on the epipolar line in the selected frame based on sum of absolute intensity differences (SAD). Once matching pixels are obtained, and then inverse depth is calculated via triangulation and its uncertainty is estimated according to the method in [6, 7].

3.3 Dense Mapping for Low-Gradient Pixels

Gradients of pixels in textureless regions are usually quite small, thus they are difficult to find corresponding matches across views via epipolar line search and in turn unable to have reliable depth estimation based on the triangulation method. We observe that most textureless regions largely consist of homogeneous planar regions. Therefore, if we could have a few pixels with depth estimation on a plane we can calculate the 3D plane parameters and in turn have depth estimation for every pixel on the plane. Based on this idea, we perform dense mapping for low-gradient pixels by first extracting candidate planar regions and then estimating plane parameters based on semi-dense map and keyframes from the previous steps.

3.3.1 Candidate Planar Region Extraction via MSCR. Several methods have been proposed for detecting candidate planar regions (CPRs) for real-time monocular dense mapping. For instance, in DPPTAM [2] the authors employed superpixels [8] computed using graph-based segmentation. However, extracting superpixels is computational expensive, i.e. over 100ms on an Intel i7-4790 processor. Additionally, due to the low repeatability of superpixels, active matching is desired to identify the true contour of each superpixel and remove unreliable superpixels which consist of non-planar objects for reconstruction. As a result, many 3D superpixels are rejected in the active matching step, yielding a low utilization of superpixels and unnecessary computational waste. To save the cost for detecting CPRs, the authors in [14] partitioned each frame into varying sized grids according to the gradients in each grid. Large grids are assumed to be homogeneous regions and small grids consist of high gradient pixels. The homogenous regions are directly matched and triangulated to have a coarse estimation. Hole-filling and linear interpolation is then applied to increase the density of coarser resolution depth map. However, finding reliable correspondence for homogeneous regions, i.e. large grids, is nontrivial.

To address the above limitations, in this paper we employed a blob region detector, named MSCR [11], for extracting CPRs. Specifically, MSCR is an ultrafast color-based affine covariant region detector with high repeatability. MSCR detect stable regions based on proximity and similarity in colour. A prune procedure based on the margin and the area size is employed for outlier filtering. MSCR can adapt to the image contours so as to occupy the entire homogeneous colour region. Compared to superpixels used in [1], CPRs detected by MSCR are much more repeatable and faster to compute, i.e. over 2X faster than using superpixel segmentation. Figure 2 (a) ~ (b) display two exemplar keyframes and (c) ~ (d) show the corresponding CPRs detected by MSCR.

3.3.2 CPR Matching. It is inevitable that CPRs detected by MSCR could contain false positives which include little planar objects (e.g.

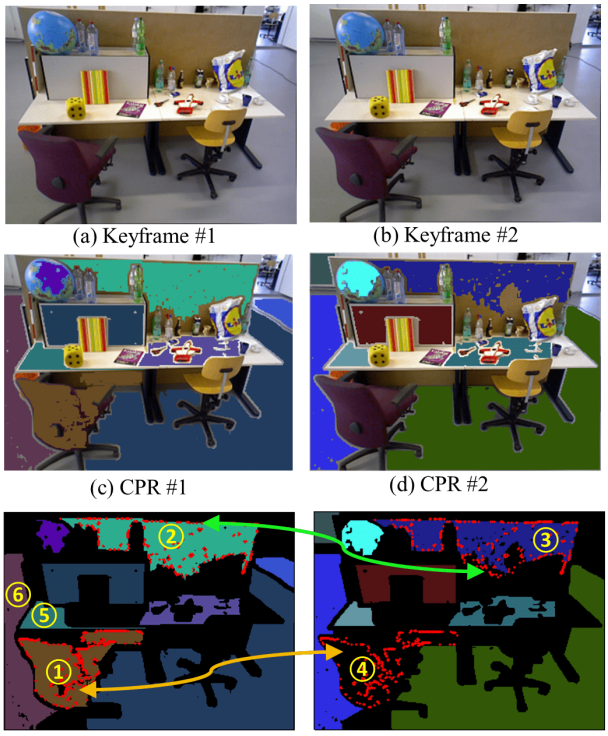

Figure 2: (a) ~ (b) show two examplar keyframes #1 and #2. (c) ~ (d) display candidate planer regions extracted by MSCR from keyframes #1 and #2. (e) illustrates the procedure of CPR matching via dual projection between consecutive keyframes. The green line and orange line denote a pair of matching CPRs and mismatched CPRs respectively.

CPR ① in the left image of Figure 2(e)). Approximating these regions using plane models could yield large reconstruction errors. Therefore, it is necessary to exclude these false positives prior the following processing. In addition, it is common that some planes which are separate in 3D coordinates are spatially adjacent 2D CPRs, such as CPRs ⑤ and ⑥, which correspond to partial table surface and floor respectively. However, for pixels which locate at the borderline of the two CPRs, it is hard to tell which CPR these pixels should belong to. Due to the limited number of pixels with depth for plane estimation, incorrect pixel assignment could yield nontrivial amount of noises and consequently lead to large reconstruction errors.

To address the above two problems, we perform CPR matching with the primary goal to remove false positive CPRs which contain little planar objects and assign semi-dense points to the correct corresponding CPRs. Specifically, we observe that the color difference for non-planar regions from two different viewpoints is greater than that of planar regions due to non-uniform shadow of non-planer objects and different reflection properties of different components of a non-planer object. Therefore, non-planer objects could be accidentally detected on a keyframe, e.g. a chair and part of the table are incorrectly detected by MSCR as a CPR on keyframe #1, while as viewpoint changes greater color differences can be seen among the back of the chair, chair cushion and the handrail, yielding disconnected components and mis-detection of a CPR on keyframe #2. In contrast, real planar objects can be much more reliably detected on both keyframes even with viewpoint changes, e.g. CPRs ② and ③. Matching CPRs across different views could also facilitate correct assignment of pixels residing at borderline. Since from some viewpoints the two adjacent CPRs could become apart, then on these keyframes the CPR assignment of pixels can be easily propagated to cases in which CPRs are spatially adjacent.

In this work, we propose an ultrafast yet effective CPR matching method via dual projection of 3D semi-dense points. We denote CPRs detected on the keyframe k as $\{m_1^k, m_2^k ... m_n^k\}$ and the semi-dense points as $\{S_1^k, S_2^k ... S_n^k\}$, where each $S_i^k (1 \leqslant i \leqslant n)$ is a 3D point set whose 2D projection to the keyframe k locate within CPR m_i^k. To search matching CPRs between two keyframes k and l, we project each S_i^k on keyframe k to another keyframe l to get a series of 2D pixels $\{\mathbf{u}^l\}$ by the projection model:

$$\{\mathbf{u}^l\} = \boldsymbol{\pi}(\mathbf{T}_k^l S_i^k) \tag{8}$$

Next, we count the number of pixels fall into every CPR in the keyframe l. The one m_j^l which possesses the most pixels and also exceeds a predetermined threshold σ is considered as a candidate match. In this paper, σ is set as half of the number of 3D semi-dense points $||S_i^k||$. Similarly, we project the corresponding semi-dense point set S_j^l of m_j^l to the keyframe k via the projection model to identify a candidate matching CPR of m_j^l on the keyframe k. If S_i^k and S_j^l are mutually matched with each other, we consider them to be a true matching pair. Otherwise, they are discarded for further processing. In our experiments, a CPR which can be reliably matched across four views is considered to be a true positive planar or piecewise planar region and is used for the subsequent dense mapping.

Figure 2(e) illustrates the CPR matching process. The red dots on CPRs ② and ③ are projected pixels of semi-dense 3D points belonging to CPR ③ to the keyframe #1 and those of CPR ② to the keyframe #2 respectively. Since majority of semi-dense points from CPR ③ can be projected into CPR ② and vice versa, we consider CPRs ② and ③ are mutually matched. If CPR ② can find mutually matched CPRs on four consecutive frames, we consider CPR ② as a true positive. For CPRs which cannot be matched to any CPRs on one consecutive frame, e.g. CPR ①, it will be directly rejected as false positives.

3.3.3 Multi-Plane Segmentation. The CPR matching step can exclude the majority of regions which consist of only non-planar objects. However, it is inevitable that some CPRs could contain multiple planes, i.e. piecewise planar regions such as the CPR ① in Figure 3(a). The red dots in Figure 3 (a) are projected pixels of 3D semi-dense points (Figure 3(b)) which locate in CPR ①. For those cases, we employ the RANSAC algorithm [9] iteratively to estimate each plane model $\Pi_1, ...\Pi_n$ progressively. Specifically, for each CPR m_i^k we iteratively select three points randomly from the corresponding semi-dense point set S_i^k. In each iteration, the selected points should not be collinear and meanwhile the pairwise

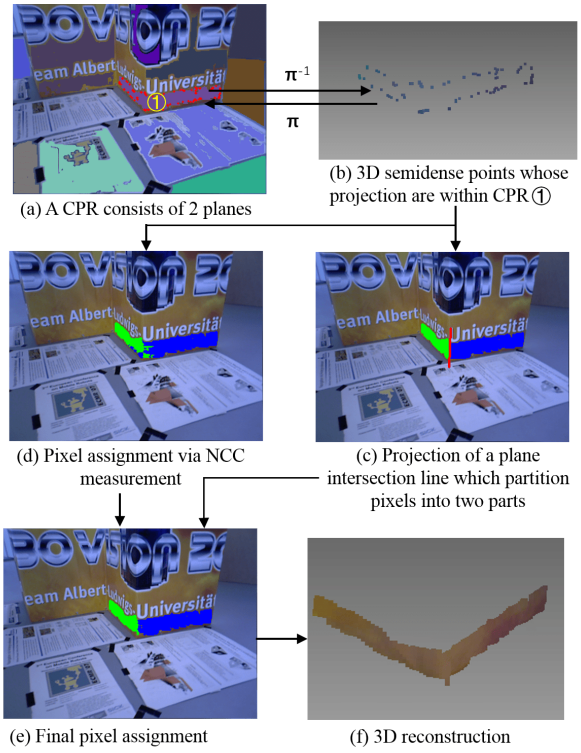

Figure 3: Multi-Plane Segmentation.

Pseudo Code1: Multi-Plane Segmentation

Input: CPRs $\{m_1, m_2, \ldots m_n\}$ and corresponding semi-dense 3D points $\{S_1, S_2, \ldots S_n\}$ in a keyframe
Output: Assign every pixel within superpixels to plane model
Procedure1: Multi-Plane Model Estimation
  for $S_i = S_1, \ldots S_n$
    $N_i = S_i.size()$
    while ($S_i.size() > 0.1 * N_i$)
      Ransac $\{S_i\}$ → plane model $(\boldsymbol{n}, d)$, inliers $\{S_{inlier}\}$;
      Remove $\{S_{inlier}\}$ **From** $\{S_i\}$
    If contain only one plane model
      for $\mathbf{u} \in m_i$ **Execute** Eq. (9)
  If more than one, **Execute Procedure2**
Procedure2: Pixels Assignment to Plane Models
  for $m_i = m_1, \ldots m_n$
    Find 3D intersection line(s) from points S_i
    Project 3D line(s) to 2D image
    Classify m_i by 2D line(s) → pixel $\mathbf{u} \in \Pi_p$
    NCC measurement → pixel $\mathbf{u} \in \Pi_q$
  if Π_p and Π_q is the same plane model, $\mathbf{u} \in \Pi_p(\Pi_q)$,
    Execute Eq. (9)
  else Discard u

Euclidean distance between the selected points should be sufficiently large. Each plane hypothesis is evaluated by the number of 3D points within a threshold distance to the plane model. Once RANSAC converges, the plane hypothesis with the most inliers is considered as the major plane Π_1. If the number of remaining outliers is sufficiently large (i.e. more than 10% of $||S_i^k||$), we remove the inliers fitting the major plane model Π_1 from S_i^k and repeat the RANSAC algorithm again in the remaining points to detect another plane Π_2. Such progress iterates until the remaining points is less than 10% of $||S_i^k||$. Consequently, for CPR m_i^k a set of n planes $\mathbf{\Pi}_i^k = \{\Pi_1, \ldots \Pi_n\}$ are obtained.

Once plane models $\mathbf{\Pi}_i^k$ are obtained, the next step is to assign every 2D pixel within m_i^k to one of the plane models. If only one plane is discovered, all 2D pixels within CPR m_i^k are assigned to this plane model $\Pi_1 = \left[\mathbf{n}_1^T, d_1\right]^T$. Accordingly, the 3D coordinates of any pixel $\mathbf{u}_{ij}^k$ in CPR m_i^k can be calculated by solving the following equations:

$$\begin{cases} \mathbf{n}_1^T \mathbf{P}_{ij} + d_1 = 0 \\ \mathbf{u}_{ij}^k = \boldsymbol{\pi} \mathbf{P}_{ij} \end{cases} \tag{9}$$

If multiple planes are discovered like CPR① in Figure 3, we perform a region segmentation based on the following two steps for pixel assignment:

- *Pixel assignment based on plane intersection line*: Given the estimated plane models, we can calculate the intersection lines of multiple planes in 3D coordinates. By projecting the 3D intersection lines to the 2D keyframe, we obtain a set of 2D lines which can act as classifiers partitioning pixels into several parts as shown in Figure 3(c).
- *Pixel assignment via Normalized Cross Correlation(NCC) measurement*: For each plane model $\Pi_j = \left[\mathbf{n}_j^T, d_j\right]^T$, we calculate its homography matrix $\mathbf{H}_{\Pi_j}$ according to Eq. (5). For each pixel $\mathbf{u}_{ij}^k$ in a CPR m_i^k on keyframe k, we project coordinates of a small patch (9×9) around $\mathbf{u}_{ij}^k$, denoted as $\mathbf{P}_{ij}^k$, to another keyframe l using all plane model-induced homography matrixes $\mathbf{H}_{\Pi_1}, \mathbf{H}_{\Pi_2}, \ldots \mathbf{H}_{\Pi_n}$, yielding $\mathbf{P}_{ij\Pi_1}^l = \mathbf{H}_{\Pi_1}\mathbf{P}_{ij}^k, \ldots \mathbf{P}_{ij-\Pi_n}^l = \mathbf{H}_{\Pi_n}\mathbf{P}_{ij}^k$. For each projected patch on keyframe l, we calculated a NCC score between the projected patch $I_l\left(\mathbf{P}_{ij-\Pi.}^l\right)$ and the original patch $I_k\left(\mathbf{P}_{ij}^k\right)$ on keyframe k according to Eq. (10)

$$NCC_{\Pi.} = \frac{\sum I_l(\mathbf{P}_{ij-\Pi.}^l) \cdot I_k(\mathbf{P}_{ij}^k)}{\sqrt{\sum I_l(\mathbf{P}_{ij-\Pi.}^l)^2 \cdot \sum I_k(\mathbf{P}_{ij}^k)^2}} \tag{10}$$

  For each pixel $\mathbf{u}_{ij}^k$, it will be assigned to the plane model Π_* which can achieve the maximum NCC_{Π_*}.

Note that each of the above steps could contain noise, in particular in contiguous regions. We combine the assignment results from both steps. That is, after processing the above two steps each pixel have two plane model assignments, as shown in Figure 3(c) and (d). If the assignment results are consistent, the pixel will be assigned to the corresponding plane. Otherwise, the assignment of the pixel is determined according to its 8-neighborhoods' assignments. In principle, applying NCC measurement across multiple views can improve the assignment accuracy, while linearly increasing the computational time. In practice, we perform this process with only one consecutive keyframe for a good trade-off between the

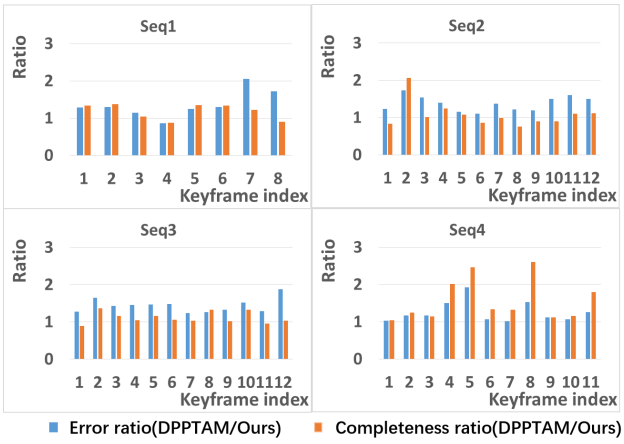

Figure 4: Reconstruction error and completeness ratios between our method and DPPTAM [2] for each keyframe.

speed and accuracy. The procedure of multi-plane segmentation is illustrated in **Pseudo Code1**.

3.3.4 Outlier Removal and Multi-view Plane Fusion. Since a piecewise planar region can be visible in several keyframes, i.e. multiple matching CPRs across views, it is necessary to obtain consistent plane models across overlapping views. However, in practice it is unavoidable that small variations (or even large differences) in plane models are derived from matching CPRs, due to variations or errors in the depth maps. To address this problem, we establish links across nearby views between mutually matching CPR-induced planes, fuse the linked planes and remove outliers whose distance is over a threshold distance to the fused plane. Specifically, as described in Sec. 3.3.2 we perform CPR matching and estimate plane model for only a CPR that can be mutually matched on four consecutive keyframes. Therefore, for each CPR m_i^k and its mutually matched CPR $m_i^{(k\pm\Delta)}$ $(\Delta = 1, 2)$, we can first link the estimated plane models $\mathbf{\Pi}_i^k$ of m_i^k with the plane models of $\mathbf{\Pi}_i^{(k\pm\Delta)}$. Such linkage can propagate to CPRs on the following consecutive keyframes. Then for each plane model Π_{ij}^k in set $\mathbf{\Pi}_i^k$, if sufficient number of points, i.e. 90% belong to Π_{ij}^k can be projected into the plane $\Pi_{il}^{(k\pm\Delta)}$, then the two planes $\Pi_{ij}^k$and $\Pi_{il}^{(k\pm\Delta)}$are linked. Once we linked all planes, for each linkage a new plane is estimated based on all 3D points belonging to the planes in this linkage. Pixels with large distance to the newly fused plane are then removed as outliers.

4 EXPERIMENT

We quantitatively evaluated the proposed method using 4 sequences Seq1: *over table* from [15] and Seq2, Seq3, Seq4 are *fr3/nonstructure_texture_near_withloop, fr3/structure_texture_far, fr2/xyz* from TUM RGB-D dataset [23], respectively. We first examine the utilization of CPRs for dense mapping by comparing the depth estimation error and reconstruction completeness of our method with DPPTAM [2] for each keyframe. A high completeness with a small depth estimation error per keyframe indicate that more CPRs can be effectively used, yielding a high CPR utilization rate and efficient reconstruction. Then, we tested the overall reconstruction error, completeness as well as runtime for each video sequence of the public datasets. Finally, we tested our system with a hand-held web camera in our own scenario and apply it to an augmented reality application. A demo video of the reconstruction results of our own scenario and AR app can be found in the supplementary material accompanying the paper. The platform we used for all evaluation is an Intel i7-4790, 3.6GHz quad core based laptop, equipped with 16 GB of RAM, running Ubuntu 14.04 operating system. We used the source code implementation provided by the authors of [2] for the evaluation of the baseline method DPPTAM.

Table 1: Average mapping time per keyframe (ms)

Dataset	[2]	Ours	Speedup
Seq1	209	72	2.9X
Seq2	174	72	2.4X
Seq3	231	85	2.7X
Seq4	190	82	2.3X

4.1 Utilization of CPR

Figure 4 displays two ratios: 1) reconstruction error ratio which is calculated as the mean reconstruction error of DPPTAM over that of our method for each keyframe, and 2) the completeness ratio which is computed as the percentage of reconstructed pixels achieved by our method over that of DPPTAM for each keyframe. We desire high completeness with small reconstruction error, thus numbers of error ratio and completeness ratio greater than 1 denote superior performance of our method to DPPTAM. From Figure 4 we observe that for most keyframes of the tested video sequences, our method achieves more accurate depth estimation and higher completeness. This result implies that the utilization of CPR is greater than that of superpixels in [2], and in turn for the same physical planes our method can reconstruct them earlier than DPPTAM and fewer keyframes are required to achieve the same completeness for our method. We believe the two main reasons for better utilization of CPR are as follows: (1) We employed MSCR which was originally designed for local feature detection for extracting CPRs. The repeatability of MSCR is much higher than the superpixel method [2] used in DPPTAM. As a result, more CPRs can be correctly matched across different keyframes and used for 3D plane reconstruction. In comparison, due to the low repeatability of superpixels in DPPTAM, many reconstructed 3D planes are removed during the active matching step since they fail to find correct matches in consecutive keyframes and in turn yield a low utilization efficiency and unnecessary computational cost. (2) We leveraged multi-plane detection and an effective classification method for assigning each pixel in a CPR to the corresponding 3D plane model. Therefore, for CPRs which contain multiple planes or even non-planar regions our method can still reconstruct subsets of pixels which locate on piecewise planar regions. In contrast, DPPTAM rejects all superpixels which consist of more than one plane, yielding a low utilization and reconstruction efficiency.

Table 2: Runtime breakdown for each step (ms)

CPR Extraction	CPR Matching	Multi-Plane Segmentation	Outlier Removal
50~55	5~8	10~15	3~5

Table 3: Relative Inverse Depth Error [%]

Dataset	[2]	[14]	Ours
Seq1	4.58	—	4.00
Seq2	1.8	6.2	1.6
Seq3	3.4	2.7	2.5
Seq4	7.8	—	7.9

Table 4: Average Keyframe Completeness [%]

	Semidense		Dense for textureless		Total	
Dataset	[2]	Ours	[2]	Ours	[2]	Ours
Seq1	38.3	37.3	10.5	14.2	48.8	51.5
Seq2	36.1	37.5	14.4	17.3	50.5	54.8
Seq3	45.1	46.4	8.8	11.4	53.9	57.9
Seq4	31.5	33.4	9.0	12.2	40.5	45.6

4.2 Overall Performance Evaluation

4.2.1 Runtime. We evaluated the average runtime per keyframe for dense mapping. Table 1 compares the runtime of our method and DPPTAM for the four public video sequences. In general, our method takes about 72 ~ 85ms to estimate depth of a keyframe, which is 2.3 ~ 2.9X faster than DPPTAM. Table 2 displays the runtime breakdown for each step. Specifically, extracting CPRs using MSCR takes the majority of the time, i.e. 50 ~ 55ms per keyframe, multi-plane segmentation takes 10 ~ 15ms and CPR matching and outlier exclusion use a negligible amount of time. The two main reasons why our system can achieve faster speed than DPPTAM are: Firstly, the CPR Extraction is fast than superpixel extraction method used in DPPTAM. Secondly, the CPR matching step filters many unstable regions prior to the subsequent reconstruction process, greatly improving the reconstruction efficiency.

4.2.2 Reconstruction Accuracy. We evaluated the reconstruction accuracy using the relative inverse depth error, a metric widely used in many SLAM and dense mapping systems [14]. Specifically, the relative inverse depth error is defined as Eq. (11):

$$E = \frac{1}{N} \cdot \sum_{i=1}^{N} \left| \frac{1/\rho_{es\_i} - 1/\rho_{gt\_i}}{1/\rho_{gt\_i}} \right| \tag{11}$$

where N represents the total number of pixels with depth estimation, and ρ_{es_i} and ρ_{gt_i} are estimated depth and groundtruth depth respectively. Table 3 compares the relative inverse depth error of our method with DPPTAM as well as alternative monocular dense mapping method [14], i.e. multi-level mapping. In general, our method outperforms the other two methods (except for Seq4: *fr2/xyz*) by achieving smaller error value. Since multi-level mapping [14] did not provide source code, thus it is difficult for us to completely duplicate their implementation. Therefore, the number of Table 3 for multi-level mapping was those reported in [14] and missing results for the Seq1: *over table* and Seq4: *fr2/xyz* is due to lack of evaluation for the two sequences in the original paper.

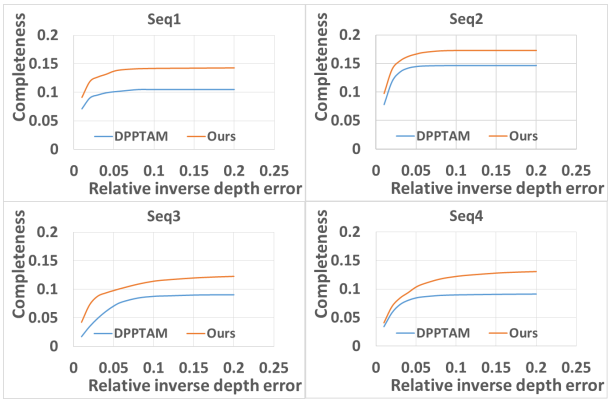

Figure 5: Completeness as a function of relative inverse depth error for the four video sequences.

4.2.3 Completeness. We plotted the completeness achieved by piecewise plane approximation as a function of relative inverse depth error in Figure 5. As can be seen, for all video sequences given the same depth error our method can produce denser map, i.e. better completeness, than DPPTAM. Table 4 provides the overall map completeness when the relative inverse depth error is smaller than 10%, including both semi-dense mapping based on pixel matching and triangulation and dense mapping using the piecewise planarity constraint, for our method and DPPTAM [2]. Results show that for semi-dense mapping for high gradient pixels, our method achieves similar completeness as [2], while for the dense mapping for textureless regions, our method outperforms [2] by ~ 4%. The total map completeness including both semi-dense and dense mapping achieved by our method is 45.6% ~ 57.9%. Figure 6 illustrates exemplar keyframes of the four videos and the final reconstruction results.

4.3 Application to Augmented Reality

We developed a simple AR application to investigate the performance of our system for practical AR tasks, in particular interactions between virtual object and real scenes with large textureless regions. Our application is "hide-and-seek" in which a virtual cartoon character is randomly initialized at a location in the physical world around the user; the user can use the hand-held camera as a "magic glass" to look for the virtual character. Once the user starts searching, our system initialises and starts to localize the camera position and reconstructs the 3D scene around the user. If the virtual character appears in the view of the hand-held camera, it can escape and hide itself in the real scene, e.g. behind a real box. A major difference of our AR app with the popular "pokemon go" is that our app enables interactions between the virtual character with the real scene, i.e. the virtual character can 'sense' the depth

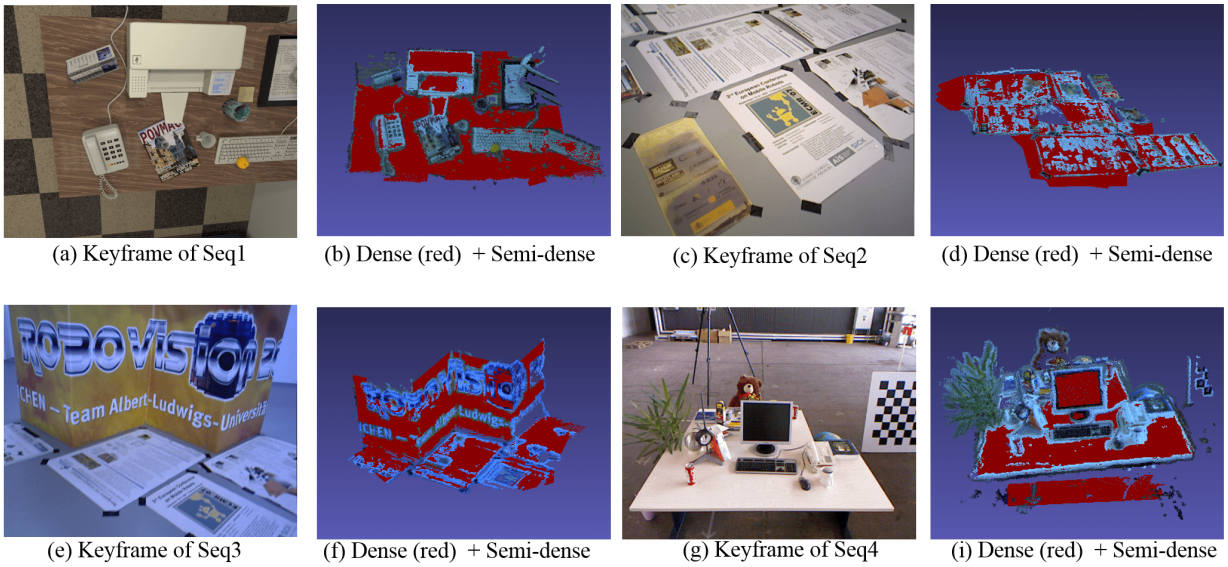

Figure 6: Exemplar keyframes of the four video sequences and the dense reconstruction obtained by our method. For the reconstructed results, pixels in red are reconstructed based on the piecewise planar assumptions and the remaining pixels are reconstructed using the semi-dense mapping method.

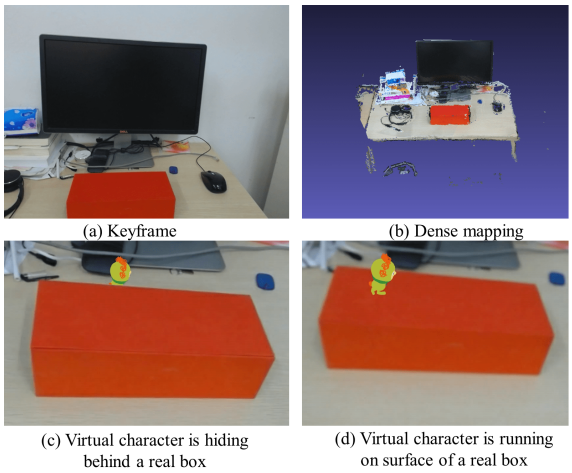

Figure 7: The system can provide dense mapping of the captured scene efficiently. The virtual character can be augmented in the real scene and interact with the physical object, e.g. hide behind the real red box or run on the surface of the box.

of real scene even regions with little texture information such as table surface, and in turn yield more immersive user experience. Some snapshots of our application is illustrated in Figure 7 and also demonstrated in the supplementary demo video.

5 CONCLUSION

In this paper, we present a novel method for real-time dense mapping based on pixel matching for highly textured regions and piecewise plane model approximation for textureless regions. Specifically, candidate planar regions are extracted by homogeneous-color region detector MSCR and then are filtered by dual projection based CPR matching. After that 3D piecewise planes are reconstructed for each CPR based on multi-plane segmentation and the corresponding semi-dense map. Multiple plane models derived from the same 3D planes across different overlapping views are linked and fused for accurate and consistent 3D reconstruction. Our experimental results show that the use of piecewise planarity approximation for textureless regions allows us to reconstruct scenes with poorly textured areas at a low computational complexity and a satisfactory accuracy. We have compared our method against the state-of-the-art monocular dense mapping method DPPTAM [2] on four public video sequences with groudtruth depth. Results show superior performance of our method to DPPTAM in terms of 2.3X~2.9X faster speed, better reconstruction accuracy and completeness. We also integrated our dense mapping method into a direct method based SLAM system and applied it to a real AR application. The live experiments show that our system can produce dense map in real-time and enable interactions between the virtual object and real scene.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China grant 61502188 and Wuhan Science and Technology Bureau award 2017010201010111.

REFERENCES

[1] Alejo Concha and Javier Civera. 2014. Using superpixels in monocular SLAM. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 365–372.

[2] Alejo Concha and Javier Civera. 2015. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 5686–5693.

[3] Amaury Dame, Victor A Prisacariu, Carl Y Ren, and Ian Reid. 2013. Dense reconstruction using 3D object shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1288–1295.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*. 2366–2374.

[5] Jakob Engel, Vladlen Koltun, and Daniel Cremers. 2016. Direct sparse odometry. *arXiv preprint arXiv:1607.02565* (2016).

[6] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*. Springer, 834–849.

[7] Jakob Engel, Jurgen Sturm, and Daniel Cremers. 2013. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*. 1449–1456.

[8] Pedro F Felzenszwalb and Daniel P Huttenlocher. 2004. Efficient graph-based image segmentation. *International journal of computer vision* 59, 2 (2004), 167–181.

[9] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[10] Alex Flint, David Murray, and Ian Reid. 2011. Manhattan scene understanding using monocular, stereo, and 3d features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2228–2235.

[11] Per-Erik Forssén. 2007. Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 1–8.

[12] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. 2014. SVO: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 15–22.

[13] Ravi Garg, Gustavo Carneiro, and Ian Reid. 2016. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*. Springer, 740–756.

[14] W Nicholas Greene, Kyel Ok, Peter Lommel, and Nicholas Roy. 2016. Multi-level mapping: Real-time dense monocular SLAM. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 833–840.

[15] Ankur Handa, Richard Newcombe, Adrien Angeli, and Andrew Davison. 2012. Real-time camera tracking: When is high frame-rate best? *Computer Vision–ECCV 2012* (2012), 222–235.

[16] Christian Kerl, Jürgen Sturm, and Daniel Cremers. 2013. Robust odometry estimation for RGB-D cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 3748–3754.

[17] Georg Klein and David Murray. 2007. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 225–234.

[18] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. 2014. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*. Springer, 703–718.

[19] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. 2016. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence* 38, 10 (2016), 2024–2039.

[20] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163.

[21] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. 2011. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2320–2327.

[22] Sunando Sengupta and Paul Sturgess. 2015. Semantic octree: Unifying recognition, reconstruction and representation via an octree constrained higher order MRF. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 1874–1879.

[23] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 573–580.