Multimedia Tools and Applications (2023) 82:42641–42659
https://doi.org/10.1007/s11042-023-15111-8

Robo-vision! 3D mesh generation of a scene for a robot for planar and non-planar complex objects

Swapna Agarwal[1] · Soumyadip Maity[1] · Hrishav Bakul Barua[1] · Brojeshwar Bhowmick[1]

Received: 1 March 2022 / Revised: 6 February 2023 / Accepted: 12 March 2023 /
Published online: 17 April 2023

Abstract
This paper presents a novel architecture to generate a world model in terms of mesh from a continuous image stream with depth information extracted from a robot's ego-vision camera. We propose two algorithms for planar and non-planar mesh generation. A Cartesian grid-based mesh fitting algorithm is proposed for mesh generation of planar objects. For mesh generation of non-planar objects, we propose a Self Organization Map based algorithm. The proposed algorithm better approaches the boundary and overall shape of the objects compared to State-Of-the-Art (SOA). Extensive experiments done on three public datasets show that our method surpasses SOA both qualitatively and quantitatively.

Keywords Computer vision · Robot vision · Telepresence · 3D vision · Reconstruction · Graphics · Cognitive vision

1 Introduction

With the advancement of Artificial Intelligence (AI) and easily available computing devices, robots are becoming usable for various complicated tasks in our day-to-day lives. Telepresence [7, 68] and teleportation robots have come to spotlight in the recent years. These robots enable a human to operate the robot from a remote location and find significant applications in remote teaching/meetings/surveillance etc. Given the current scenario of the COVID-19

S. Agarwal and S. Maity have contributed equally to the paper.

✉ Swapna Agarwal
agarwal.swapna@gmail.com

Soumyadip Maity
soumyadip.maity@gmail.com

Hrishav Bakul Barua
hrishav.smit5@gmail.com

Brojeshwar Bhowmick
b.bhowmick@tcs.com

[1] Robotics & Autonomous Systems group, TCS Research, Kolkata, India

pandemic, telepresence robots are in much demand for hospital setups to mitigate the physical communication of medical staff with corona patients. In parallel, several applications of digital-twin are in high demand after the publication of Industry 4.0, a holistic automation and smart manufacturing execution architecture. For example, the images captured in the production line can directly be utilized to generate a digital replica of the environment. This replica can be passed to the product designer so that he/she can provide expert guidance on improving the production process. These applications require a good structural representation of the scene for remote assistance and guidance. XR-Aided Industry i.e. virtual reality, augmented reality, and mixed reality technologies also need the 3D reconstruction for base environment creation. The paper [21] beautifully reviews the importance of 3D scene generation for telepresence, teleoperation and XR technologies.

As the robotics applications demand, the reconstructed 3D environment should not be too sparse to understand the occupancy, it should not be too heavy to transfer over the network for teleoperations/telepresence applications, and at the same time the geometry of the reconstruction should be preserved properly. For high-fidelity scene reconstruction, often LIDAR sensors [62, 63, 65, 73] are used. But these solutions are too costly. Due to the size constraint, these sensors are often not flexible to mount on smaller robots/drones. At the same time, the output point-clouds contain redundant points and are not suitable to transfer over the network due to higher disk size. RGB image based reconstruction methods work well for constraint static environments but miserably fail in texture-less / less-textured environments. The results generated by these methods also suffer from depth inconsistency. Recently, wide availability of cheaper and smaller depth controllers like Microsoft's Kinect [45], HoloLens [32], Intel RealSense[46] encourage researchers to include depth images with RGB [10, 17, 76, 77] for better consistent 3D reconstruction. In this work, we mainly focus to design a novel pipeline to model the world for recent robotic applications given a set of input RGBD images of the environment.

In our proposed pipeline, we mostly exploit planar properties of the scene for pose estimation as well as planar mesh creation. Compared to point features or volumetric representation, the usage of higher-level features like planes always help to preserve the structural property of the scene and thus, increase the accuracy of the pipeline. But a combination of point and planar properties can yield even better results [29, 39, 74, 78, 80]. We segment and model planes with irregular boundary shapes or holes. We further process the non-planar arbitrary objects of the scene to generate a triangular mesh that keeps the structural properties intact. We incrementally refine those over time to form a high-quality mesh of the environment. We represent the world model by triangular meshes as maintaining the point-cloud is always complex in terms of both time and space. To generate mesh from a set of map points, the majority of the State-Of-the-Art (SOA) systems use Delaunay Triangulation [9], but suffers from "curse of dimensionality" [3] i.e. the time complexity of the system increases exponentially in a higher dimension or with a higher number of points. In our paper, we propose two different mesh generation algorithms which are simpler and unlike Delaunay Triangulation, work really well in higher dimensions too.

We design and develop a novel pipeline for generating a world model represented in terms of mesh for robotic applications. The major contributions of this paper are three-fold.

- We exploit the point and planar properties of the scene for tracking and mesh generation of the planes. At the same time, we process the non-planar arbitrary objects for completeness of the scene.
- For non-planar arbitrary objects, we propose a Self Organizing Map (SOM) [36] based novel mesh generation method, which is more accurate, robust and better represents the object boundaries than the state-of-the-art (SOA) mesh creation methods.

- We propose a simple grid fitting algorithm for planar surfaces, which is less complex compared to the SOA.

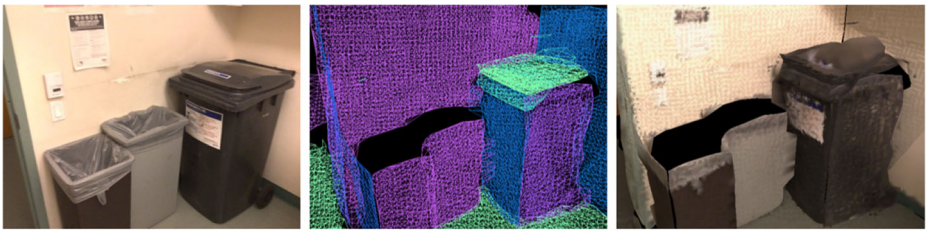
We present our pipeline along with the proposed surface fitting algorithms in Section 3. The results are presented and compared with SOA in Section 4. The conclusions and future work are stated briefly in Section 5. We place our contribution with respect to the related SOA in the next section.

2 Related works

3D reconstruction is a well-researched topic for the last two decades and is popular in both the robotics and computer vision communities. Many research papers are published on this topic including both geometric [35] and learning-based [33] approaches. Although, due to the complex layout, occlusions, and complicated object interactions, indoor 3D reconstruction still poses challenges [34, 44]. Structure from Motion (SfM) [87] is one of the main approaches for geometric reconstruction. The existing methods in the SfM literature [5, 14, 23, 75] suffer from computationally intensive steps like feature extraction and matching, mainly due to the usage of the point features. Unlike those, we use the plane as the feature and match the planes between consecutive keyframes. A plane consists of a set of points, thus the matching time is significantly reduced as the number of extracted features per frame are significantly less.

Over the last few years, another similar research topic on SLAM (Simultaneous Localization And Mapping) [25, 28, 47, 51, 58, 59] has progressed a lot, but world model creation with high fidelity objects still requires potential research. Although many methods [12, 16, 19, 28, 30, 40, 43, 54, 66, 67, 69, 71] have been proposed in this area in recent times, most of those work for offline reconstruction and not suitable for our application. World model generated by other papers [16, 30, 66, 71] either does not preserve the structural property of the environment or is too sparse to understand the environment. Our method applies specialized approaches for non-planar objects and planar objects separately. This results in a reconstruction which is a combination of dense (for non-pla ar objects) and sparse (for planar objects) mesh, good enough to preserve the structural properties of the non-planar and planar objects. Figure 1 shows the mesh generated by our pipeline on a publicly available dataset [16].

Several deep-learning based approaches are present in the SOA that reconstruct the environment in near real-time. As an example, the paper [38] discusses a very interesting method using a learning model to construct textured 3D shapes with precise geometry from 2D images. Generally supervised learning models [52] would require training on a huge labeled dataset of textured/colored meshes which is not easy to achieve. To overcome these prob-



(a) (b) (c)

Fig. 1 3D Mesh generated by proposed pipeline on Copyroom [16] dataset. (a): original frame, (b): generated normal wire-frame, (c): vertex-colored 3D mesh

lems, we propose an unsupervised learning based novel pipeline to generate a 3D mesh of the environment from an ego-centric RGBD sequence. Two interesting surveys on 3D object reconstruction under the umbrella of deep learning are presented in [33] and [26]. Recently, Mildenhall et. al. has proposed Neural Radiance Field (NeRF) [55] as a fully-connected neural network, which is able to generate novel views of complex 3D scenes. NeRF takes 2D observations as input and represents the environment as an implicit surface. Many follow-up papers [6, 18, 31, 83, 85] incorporate depth with RGB and demonstrate very good reconstruction. The major drawback of these methods is the higher training time due to the global (non-incremental) reconstruction strategy. Thus, these methods are completely inappropriate for onine robotic applications. Our method exploits the topological ordering capability of unsupervised Self-Organizing-Map (SOM) neural network. Our proposed method does not need a training phase and approximates complex real objects in few iterations.

Some SOA methods represent each non-planar object in the scene with a predefined model such as CAD. The problem with such methods is getting a such huge number of models, which is infeasible for real-life applications. Therefore, we need a more generalized method. Some methods [42, 48, 82] in the literature use SOM-based surface fitting for such purpose. We propose a SOM-based generalized method for generating the mesh, fitting the surface of the objects. Our method better maintains the structure and better represents the boundary compared to SOA. The next section, presents the overall design of our proposed approach.

3 Our system: overview and design

Our approach exploits the point as well as planar properties of the scene for pose correction as well as planar mesh creation. Structural reconstruction of the environment often demands a higher level of features for better accuracy, at the same time point features can give more granular information to achieve high fidelity. So, we choose points and planes both in dense volumetric representations for feature matching. We use an RGBD sensor's data (from the robot's ego view) as input. We process the RGB data to segment the planes and further enhance the result using depth information. First, we use the plane segmentation results to estimate the plane parameters in Hessian/Hesse Normal Form (HNF) [8]. Later, we use and compare two approaches for plane tracking and pose estimation. In the first approach, we take the plane parameters in HNF and convert it to Plane Parameter Space (PPS) [72]. Based on the plane parameters in PPS, we match the planes between two consecutive frames and estimate the relative pose between the frames. The other approach exploits the point and planar properties both as described in Plane Bundle Adjustment (PBA) [84]. We further separate the non-planar points and form a point-cloud accumulating those points in consecutive frames using estimated pose. To reduce redundancy and process the pipeline faster, we limit our mesh creation on some selected frames, also known as keyframes. For mesh creation, we adopt two different approaches. Firstly, we use a surface approximation-based mesh creation algorithm on newly segmented planes in every keyframe. Whereas, for non-planar areas, we use the point-cloud to create mesh using our novel SOM-based approach. The neural network (SOM) based approach provides a highly parallelizable method (described in Section 3.5) for fast output [60]. We update the created meshes incrementally using the estimated pose from next keyframe onward. Figure 2 depicts the overall pipeline of our proposed method. Next, we describe the major components of our simple yet robust pipeline.

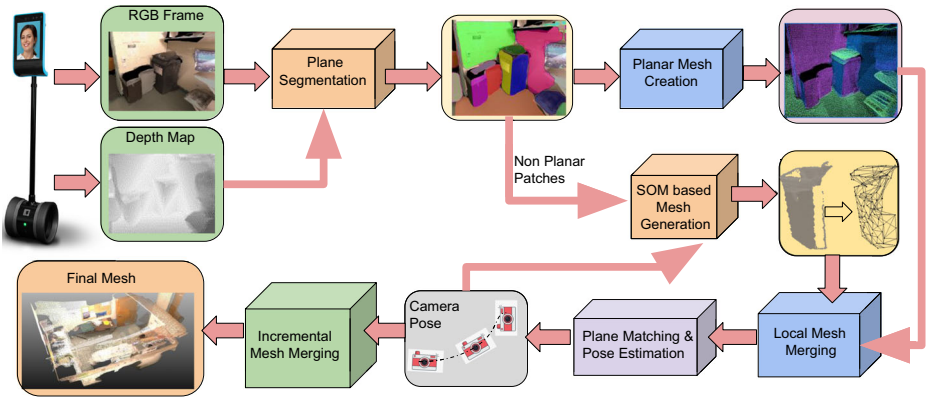


Fig. 2 The block diagram of our proposed pipeline of robust 3D scene reconstruction from the robot's ego view (RGBD sensor)

3.1 Plane segmentation

We adopt the concept of Plane RCNN [49] that uses Mask RCNN [37] for plane detection and segmentation. We choose Plane RCNN as it is capable of handling any number of planes, whereas other SOA plane segmentation pipelines like PlaneNet [50] and PlaneRecover [81] need prior information about the maximum number of planes. Plane RCNN segments only two categories, either "planar" or "non-planar". We adopt the same concept and use the Mask R-CNN pre-trained on ScanNet [15] dataset. We further refine the plane boundaries using the depth information. Once the plane segmentation is done, we calculate the plane parameters using a RANSAC [24] in HNF, where a plane π is represented by $[q_\pi, d_\pi]$. Here q_π is the unit normal vector $[q_x, q_y, q_z] \in IR^3$ on the plane and $d_\pi \in IR$ is the distance of π from the origin. As we are using a less noisy depth information coming from a depth sensor, RANSAC works really well with less number of iterations. After plane segmentation, we need to track planes and estimate pose.

3.2 Plane tracking and pose estimation

For plane Ttracking, we adopt the PPS [72] based plane representation for frame to frame plane matching proposed by Sun et al.. A plane $\pi = [q_x, q_y, q_z, d_\pi]$ in Cartesian coordinate is transformed in PPS by using the (1).

$$\begin{cases} \theta_\pi = arccos(q_z) \\ \varphi_\pi = atan2(q_y, q_x) \\ d_\pi = d_\pi \end{cases} \tag{1}$$

Every Cartesian plane is represented by a point in PPS and all the Cartesian points on the plane lie on the same point in PPS. We exploit this property to find the plane correspondence between two consecutive frames. Based on the correspondence, this algorithm estimates the relative pose R, t between two consecutive frames. Further, we use a plane based bundle adjustment to refine the poses as described in next sub-section.

3.3 Pose refinement

Bundle adjustment [2] is a commonly used optimization technique for camera pose (position and orientation) estimation. But the accuracy of this method highly depends upon the number of feature points and the accuracy of the correspondences. On the other-hand, planes are the high level geometric primitives of a scene and help to reduce geometrical ambiguity caused by noisy feature points. We already have plane parameters from previous section, we exploit those parameters along with plain points to estimate the camera pose. Zhou et. al. [84] propose a Planar Bundle Adjustment technique (PBA) that formulates the optimization problem using a point-to-plane cost function for pose estimation. In PBA, both the plane and the points and their relationships are considered for pose estimation. At the same time, if the number of plane points is large, it can lead to a large scale least-squares problem which is efficiently addressed in [84] itself. Here, the squared distance between a plane and plane points is used as the cost function.

Let us assume, R_i and t_i are the rotation and translation respectively to transform ith frame from local coordinate system to a global coordinate system. Also assume $p_{ijk} \in IP_j$, where IP_j is the set of K number of points which are sampled from ith frame and lie on j-th plane π_j. Note that there may exist multiple planes in one frame. The signed distance between a point p_{ijk} and corresponding plane π_j can be written as (2).

$$\delta_{ijk} = [q_x, q_y, q_z]_j.(R_i\, p_{ijk} + t_i) + d_{\pi j} \tag{2}$$

Here, '.' represents a dot product and δ^2_{ijk} is the point-to-plane cost. We use PBA to perform a motion only optimization to get a refined camera pose $[R_i, t_i]$ by minimizing the point-to-plane cost. In the next sub-section we discuss the 3D mesh creation technique.

Require: q_π, d_π, grid dimension d_{grid}, plane points $[P]$

1: Calculate rotation R_π between q_π and z-axis A_z
   $R = I + [v]_X + [v]^2_X \frac{1}{1+C}$
   Where, I is the Identity Matrix,
   $v = A_z \times q_\pi$
   $C = A_z \cdot q_\pi$
   $[v]_X$ is the skew-symmetric matrix of v
2: Rotate all points $[P] \in IR^3$ on π using R_π
   $[P]_{rot} = R_\pi[P]$
3: Now π become parallel to xy plane and we simply use x, y coordinate of $[P]_{rot}$ to project the plane in IR^2
   I_π = Project($[P]_{rot}$)
4: Create a diagonal grid $G \in IR^2$ with dimension d_{grid} of each grid
5: Fit the grid G on I_π
6: **for all** Triangle $g \in G$ **do**
7: **if** g is 75% fill with Foreground pixels **then**
8: Keep g
9: **else**
10: Discard g
11: **end if**
12: **end for**
13: END

Algorithm 1 Generate 3D mesh for plane $\pi = [q_\pi, d_\pi]$.

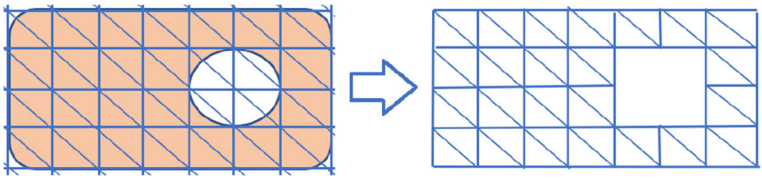


Fig. 3 Planar mesh generation using diagonal grid

3.4 3D mesh propagation for planes

Delaunay triangulation [9] is a well-known technique to generate triangular meshes in IR^2, but its complexity is high with large number of points [3, 53]. Thus, it is not suitable for finer mesh creation. So we avoid it and devise a Cartesian grid based mesh fitting algorithm as our own contribution as described in Algorithm 1. By changing the grid dimension represented by d_{grid} we can control the generated mesh quality from coarse to finer. Figure 3 shows a sample input and output of our algorithm. While Algorithm 1 is suitable for faster meshing of planes, for non-planar objects we need a non-planar generalized method. Next, we present our proposed method for generating meshes for non-planar objects.

3.5 3D mesh creation for non-planar objects

Given a non-planar object point-cloud extracted from a sequence of RGBD images, we need to approximate the object surface with a mesh. For getting this object point-cloud, we do the following. Given the RGB image, we apply object segmentation method such as Mask RCNN [1] that provides the mask of the object. Using the mask, from the RGBD image, we segment out the point-cloud of the object. Figure 4 illustrates this with an example.

We compare the mesh with a fishing net with fixed number of knots with elastic edges. We cover the entire object (point-cloud) with the fishing net by stretching its nodes. The amount and direction of this stretch is directed by the distribution of the points in the object.

Given the topology-preserving universal approximation capability of Kohonen's Self Organizing Map (SOM) [36], we adopt a two-dimensional lattice (for understanding see Fig. 5(a)) of SOM as our base method for mesh approximation.

Let us assume that the point-cloud of the given object consists of l points represented by $X = \{\mathbf{x}_i \in R^3\}$, $i = 1, .., l$, the 2D lattice consists of $M = m \times m$ nodes where the set of nodes is represented by set of weight vectors $W = \{\mathbf{w}_j \in R^3\}$, $j = 1, .., M$. The $\mathbf{x}_i$ acts as the activation pattern. Given $\mathbf{x}_i$, the nodes $\mathbf{w}_j$ compete among each other. The winning node $i(\mathbf{x})$ learns the pattern of $\mathbf{x}_i$ following a learning rate $\eta(n)$ and cooperates with the neighboring nodes for learning. The neighborhood is defined by $h_{j,i(\mathbf{x})}(n) = \exp(-d^2/2\sigma^2(n))$, where d is the Euclidean distance between $i(\mathbf{x})$ and jth node, σ is the spread of the neighborhood function and n is epoch number. A number of epochs N arranges the nodes of the 2D lattice such that it covers the surface of the object. The base SOM algorithm is given in [36].

Since, SOM [36] is an unsupervised clustering algorithm, a set of input points (x_i) are represented by a central mesh node (w_j). Thus, general SOM algorithm does not extend the mesh to the exact boundary of the input point-cloud and does not represent the boundary well (see Fig. 5). Therefore, we propose to modify the conventional SOM algorithm

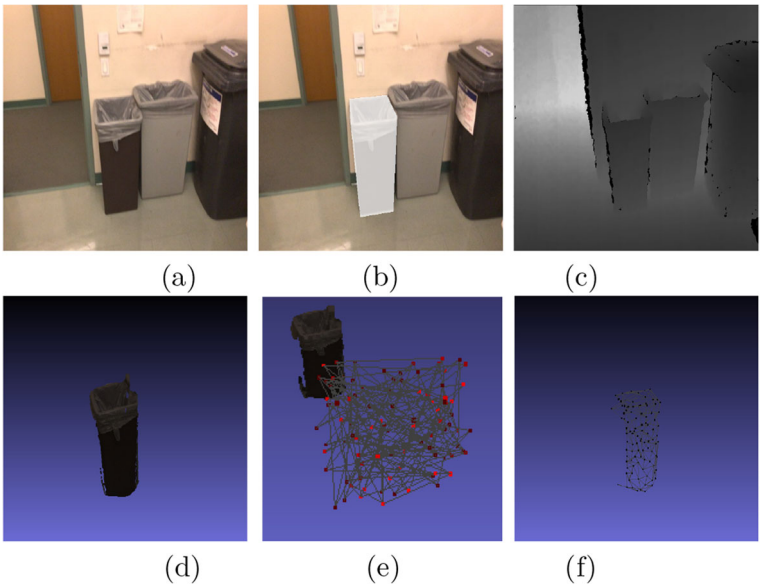


Fig. 4 Different steps of non-planar mesh approximation given an RGBD image. (a): Input RGB image of the scene, (b): object segmentation, (c): depth map, (d): point-cloud of the object extracted from the scene RGB of (a), segmentation mask of (b) and depth map of (c), (e): initialization of the mesh nodes and (f): approximated mesh

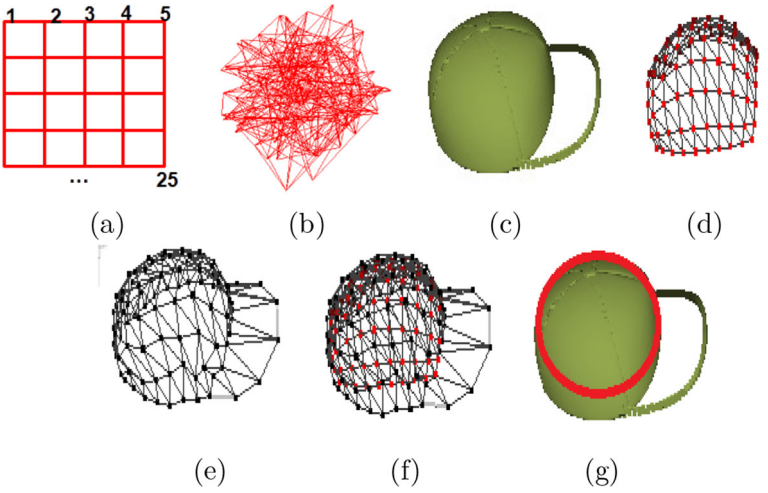


Fig. 5 Showing boundary effect due to conventional SOM and the proposed SOM. (a): Schematic diagram depicting a 2D 5×5 lattice of mesh nodes (the numbers represent the node index), (b): condition of 10×10 mesh after initialization, (c): input cap point-cloud, (d): result of applying conventional SOM, (e): result of applying proposed Algorithm 2, (f): (e) overlayed on (d), (g): (d) and (e) overlayed on (c), the red curve shows the boundary of (d)

Apply N_1 iterations of conventional SOM [23] on the input point-cloud X to generate a mesh having M nodes.
Find the set XB of boundary input points $\mathbf{xb} \in XB$ such that the winning node $i(\mathbf{xb})$ is a boundary node. $XB \subset X$. $i(\mathbf{xb}) = \arg\min_j ||\mathbf{xb} - \mathbf{w}_j||$, $j = 1, ..., M$.
for $n = 1, .., N$ **do**
    $n = n + N/2$. /* initiate the $\eta(n)$ and $\sigma(n)$ with decreased value.*/
    for $p = 1, ...l$ **do**
        Sampling: Randomly draw a sample say, $\mathbf{x}$ from X.
        Find the winning node $i(\mathbf{x})$ using the minimum distance Euclidean criteria. $i(\mathbf{x}) = \arg\min_j ||\mathbf{x}(p) - \mathbf{w}_j||$, $j = 1, ..., M$.
        if $\mathbf{x} \in XB$ **then**
            $\sigma(n) = \sigma(n)/2$. /* To decrease the elasticity of the boundary nodes.*/
            $\eta(n) = \eta(n) \times 2.5$. /* To increase the flexibility of the boundary nodes.*/
            Updating: Update the weights of all the nodes using the following formula. $\mathbf{w}_j(p+1) = \mathbf{w}_j(p) + \eta(n) h_{j,i(\mathbf{x})}(n)(\mathbf{x}(p) - \mathbf{w}_j(p))$. $j = 1, ..., M$ Here, $\eta(n)$ and $h_{j,i(\mathbf{x})}(n)$ are learning rate parameter and neighborhood function centered at the winning node $i(\mathbf{x})$. Both these parameters decay with time.
            $\eta(n) = \eta(n)/2.5$.
            $\sigma(n) = \sigma(n) \times 2$.
        else
            Updating: Update the weights of all the nodes using the following formula. $\mathbf{w}_j(p+1) = \mathbf{w}_j(p) + \eta(n) h_{j,i(\mathbf{x})}(n)(\mathbf{x}(p) - \mathbf{w}_j(p))$. $j = 1, ..., M$.
        end if
    end for
    $\eta(n) = n^{-0.2} * \xi(n)$ where $\xi(n) = 1 - \exp(5(n - N)/N)$.
    $\sigma(n) = \sigma N + \xi(n)(\sigma 0 \times 0.05^{n/N} - \sigma N) \times n^{-0.25}$.
end for

Algorithm 2 Algorithms to generate a mesh addressing boundary condition given a point-cloud.

[36] such that the mesh nodes near the boundary have greater learning rate and less elasticity. As a result, the boundary nodes move freely, almost independent of their neighboring inner nodes, towards the input points on the boundary. Thus, the resulting mesh covers the boundary of the object well.

The existing literature [36] partitions the entire SOM iterations into two phases: (1) ordering, and (2) convergence. We look at SOM iterations as consisting of three phases: (1) unfurling phase where from the state of complete un-organizations (Fig. 5(b)), SOM mesh unfurls itself, (2) ordering phase where the unfurled mesh approximates the point-cloud surface and (3) convergence phase where the ordered mesh reaches out to the boundary. For accelerated result, in the unfurling phase, both the learning rate η and the spread represented by σ of neighborhood should be very high initially. Then they both should decrease very quickly so that the unfurling process happening in the first iteration should not get undone in the subsequent iterations. For similar reason, in the convergence phase the learning rate should decrease very fast. The σ value should also be very small in the ordering phase so that the boundary nodes of the lattice can approach the boundary points of the object and does not affect already ordered inner points of the lattice. The decay functions of η and σ in conventional SOM [36] do not satisfy these requirements. Equations (3) and (4) present

the conventional decay functions and (5) and (6) present the modified decay functions that satisfy our need.

$$\eta(n) = \eta_0 \exp(-n/\tau_1), n = 1, 2, .., N.\tau_1 = N \tag{3}$$

$$\sigma(n) = \sigma_0 \exp(-n/\tau_2), n = 1, 2, .., N.\tau_2 = N/\log\sigma_0. \tag{4}$$

$$\eta(n) = n^{-0.2}\xi(n), n = 1, 2, .., N. \tag{5}$$

$$\sigma(n) = \sigma N + \xi(n)(\sigma 0 \times 0.05^{n/N} - \sigma N)n^{-0.25}, n = 1, 2, .., N. \tag{6}$$

Here, $\xi(n) = 1 - \exp(5(n-N)/N)$, $\sigma 0$ and σN represent the desired spread of the neighborhood function at iteration 0 and iteration N. At the start of iterations, we want that almost all the nodes must be affected by the update. Since, we are considering a 10×10 lattice, we set $\sigma 0 = 5$. At the end of iterations, we want that the winning node does not affect its neighborhood, thus we set $\sigma N = 0.5$. The two constants 2.5 and 2.0 in steps 9, 10, 12 and 13 of Algorithm 2 are set empirically. Figure 6 compares the modified vs. the conventional decay functions for η and σ.

By the end of ordering phase both the η (learning rate) and σ become low. In the convergence phase, we want the boundary nodes to have higher η so that they can move towards the input boundary points. We want them to have smaller σ (less elasticity) so that they do not make huge alteration to the already arranged inner nodes. We propose Algorithm 2 as our novel contribution that satisfies the above mentioned requirements and addresses boundary condition. Note that the main update at steps 11 and 15 of Algorithm 2 can be implemented in a parallel fashion for all the nodes in the lattice. This helps in acceleration of algorithm execution.

For further refining the mesh shape in local areas having sudden change in normal, we do the following. After running Algorithm 2, for each node $\mathbf{w}$, we find the set $X' = \{\mathbf{x}_i, i = 1, ..., r : \mathbf{x}_i \in X, \mathbf{w} = \arg\min_j ||\mathbf{x}_i - \mathbf{w}_j||, j = 1, ..., M\}$ Then we get the average using $\mathbf{x}' = \sum_{i=1}^{r} \mathbf{x}_i / |\mathbf{x}_i|$. If the Euclidean distance between $\mathbf{w}$ and $\mathbf{x}'$ is greater than a threshold, we apply ARAP [41] deformation keeping $\mathbf{w}$ as source and $\mathbf{x}'$ as destination. One example is shown in Fig. 1 in the supplementary material. Next, we evaluate the efficacy of our approach.

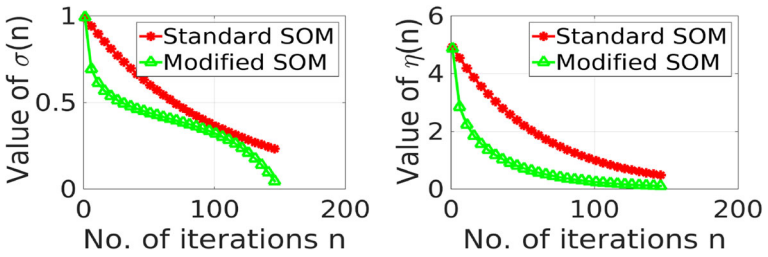


Fig. 6 Comparing the modified vs. the convention decay functions for σ (left) and η (right). Notice that during the initial and towards the end of iterations the modified σ decreases faster. The value of modified η is also less in the ordering phase compared to the conventional one

4 Experimental results and evaluation

4.1 Datasets and setups

We use three known datasets to evaluate our pipeline and the individual components. The ScanNet [15] datset is used for plane segmentation part. For non-planar objects, we also use ShapeNet [11]. Finally for the entire pipeline evaluation we use, Apt0 [4], Copyroom [13] and Office3 [61] datasets. All the experiments are done w.r.t. a Double2 [20] robot mounted with a RealSense [64] camera for RGBD data. The pipeline is run on an edge device connected to our robot having single-core CPU and 6GB RAM.

4.2 Results on 3D mesh creation for non-planer objects

To test the efficacy of the proposed Algorithm 2, we have done extensive experiments, both qualitative and quantitative, on two types of data. The first type consists of object point-clouds extracted from real RGBD images (Copyroom and Office3 datasets). The second type consists of object point-clouds taken from CAD models (ShapeNet dataset [11, 22]). Whereas, the point-clouds from CAD model are expected to be ideal with no noise or hole, the point-clouds extracted from RGBD images are expected to contains noise, holes and uneven density of points. Our algorithm smoothens noise, fills the holes and works well for uneven density. The run time of our proposed algorithms is proportional to the size (l) of the input point-cloud. Therefore, to reduce the size while preserving the shape and topology of the input point-cloud, we apply voxel grid filter. This filtering also helps us to tackle the issue of uneven density of input point-clouds. The resolution of the generated mesh is determined by the number $M = m \times m$ of nodes in the approximating 2D lattice. Greater the value of M, greater is the resolution of the generated mesh. The upper limit of this resolution is restricted by the resolution of the input object point-cloud. We select this M such that the generated mesh is dense enough to represent the structure of the object and sparse enough for low cost storage. Tables 1 and 2 show some results on real and CAD objects. We also compare our results with that of the SOA [70] which also utilizes SOM for approximating object surface. Comparing the third and the fourth columns of Tables 1 and 2 we observe that our method better approximates the surface shapes of the input objects. Column five overlays the result of [70] on that of ours. These results in column four shows that our method stretches the mesh boundary to the object boundary better as compared to [70]. The first row of Tables 1 and 2 shows one example where our method can be seen to fill the hole.

In Tables 3 and 4 the two quantitative metrics (as in [27]) are defined as follows. Accuracy: distance d such that a given percentage of the reconstruction is within d from the ground truth model. Completeness: percentage of the ground truth model that is within a given distance from the reconstruction. For accuracy, we define given percentage as 90%. Different point-clouds can be in different scales. Therefore, for completeness, we define given distance as one tenth of the maximum distance between any two points along x, y and z axis. Note that smaller the value of accuracy, the better it is. Similarly, bigger the value of completeness, the better it is. From Tables 3 and 4 we can say that our method generates better result compared to SOA [70] both qualitatively and quantitatively.

For the sake of completeness and to have an idea of the overall performance of our approach on the whole datasets, we have done two sets of experiments. In the first set, we have taken the non-noisy point-clouds of 30 objects from three different sources [56, 57, 86]. In the second set, we have extracted (segmented) the point-clouds of the objects

Table 1 Qualitative comparison of our method with [70] for object point-clouds extracted from real RGBD images

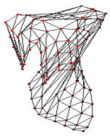
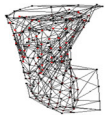
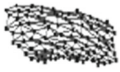
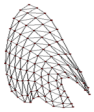
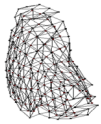
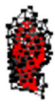
Obj.	Input point-cloud	Our mesh	Mesh by [70]	[70] on our
Dustbin				
Keyboard				
Chair back				
Bottle				
Bottle				

Table 2 Qualitative comparison of our method with [70] for object point-clouds from CAD models

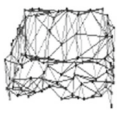
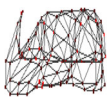
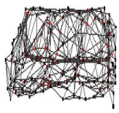
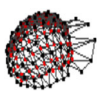
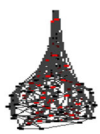
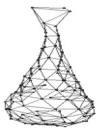
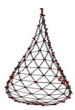
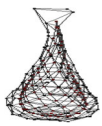
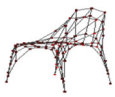
Obj.	Input point-cloud	Our mesh	Mesh by [70]	[70] on our
Sofa				
Cap				
Lamp				
Vase				
Chair				

Table 3 Quantitative comparison of our method with [70] for object point-clouds extracted from real RGBD images

Obj.	Input point-cloud	Acc. Our↓	Acc. [70]↓	Com. our↑	Com. [70]↑
Dustbin		**0.0375**	0.066	**99**	92
Keyboard		**0.013**	0.026	**100**	91
Chair back		**0.021**	0.027	**100**	99
Bottle		**0.009**	0.02	**99**	88
Bottle		**0.009**	0.023	**100**	80

Acc.: Accuracy, Com.: Completeness

Table 4 Quantitative comparison of our method with [70] for object point-clouds from CAD models

Obj.	Input point-cloud	Acc. Our↓	Acc. [70]↓	Com. our↑	Com. [70]↑
Sofa		**33.91**	49.51	**99**	92
Cap		**5.03**	8.26	**98**	89
Lamp		**24.53**	42.54	**100**	**100**
Vase		**161.02**	234.55	**99**	96
Chair		**44.19**	101.24	**99**	81

Acc.: Accuracy, Com.: Completeness

Table 5 Quantitative comparison of our method with [70] for point-clouds of synthetic objects from [56, 57, 86] and noisy point-clouds extracted from Copyroom3 indoor scene dataset [13]

Input point-cloud	Acc. Our↓	Acc. [70]↓	Com. our↑	Com. [70]↑
Copyroom3 [13]	**1.79**	3.24	**99.6**	90.0
[56, 57, 86]	**9.1**	11.2	**99**	91.6

Acc.: Accuracy, Com.: Completeness

from the indoor scene of the Copyroom3 dataset as explained in Section 3.5 and also in Fig. 4. Since these object point-clouds in the second set are processed from RGBD images of indoor scenes, these point-clouds contain noise. In these two set of experiments, we have run our Algorithm 2 and also the approach by Steffen et. al., [70] on the objects from the whole of two datasets. The results are compared in terms of two quantitative measurements namely accuracy and completeness (defined in Section 4.2). The results are presented in Table 5. Since the volume of different objects are different, instead of mentioning the absolute distance d for which 90% of the reconstructed points are within d, we have mentioned accuracy as $d \times 100/$(maximum distance between any two points in that object). Table 5 shows that our method is better in terms of both accuracy and completeness. The modified Algorithm 2 for better boundary approximation, especially the two (5) and (6) are the reason behind this improvement with respect to the SOA.

4.3 Final mesh results

The world model containing planar and non-planar object meshes generated by our novel pipeline holds the structural property of the environment and thus perfect for remote virtual navigation and manipulation. Figure 1(c) shows the reconstructed environment mesh. The mesh normal in Fig. 1(b) clearly shows a smooth output as desired. Figure 7 shows the complete generated mesh by our proposed pipeline on three different publicly available datasets. Some other results are presented in the supplementary material.

4.4 Execution time

Plane extraction, plane matching, planar and non-planar mesh generation are four computational building blocks of our pipeline. As we design our system to process only keyframes and the Double2 robot movement speed is not on the higher side, we need to process 2-3 keyframes per second. For plane extraction, we use Plane RCNN, which is a learning-based

Fig. 7 Vertex colored 3D mesh generated by our pipeline on (a): Office3, (b): Apt0, (c): Copyroom

plane segmentation algorithm and generates output in near real-time [79] in our experimental setup. For plane matching, we wisely exploit PPS-based plane representation and are able to track the planes in real-time. For planar mesh generation, instead of using computationally expensive Delaunay triangulation, we use a much faster grid fitting algorithm. Any indoor environment mostly has planar objects like walls, floors, and ceilings. So, the usage of a less expensive mesh generation algorithm for the most part of the environment helps to minimize the run-time of the pipeline. For non-planar regions, we modify the SOM-based clustering algorithm, which is highly parallelizable. The main updates of steps 11 and 15 of Algorithm 2 can be done in parallel for all the M nodes of the lattice. Thus, can be implemented easily in multi-threaded CPU / GPU. We have implemented the customized SOM to run in several threads in a CPU-based Edge device. In the CPU-based edge device, our whole pipeline is able to run at 3 frames per second, which is sufficient for a keyframe-based pipeline (3 key-frames per second) in a constrained indoor environment. Utilization of GPU will increase the speed further.

5 Conclusion and future works

We have proposed a complete pipeline for the representation of a 3D world model in terms of triangular meshes given a continuous sequence of RGBD images of the scene captured from the robot's ego view. Our method being simplistic, can be easily integrated into any robotic platform. The method does not have any training phase and does not require a huge training set as required by supervised learning based methods. Thus, this method is generalized to any indoor environment. Our mesh approximation of the objects in the scene better represents the shape and size of the object compared to SOA. Both qualitative and quantitative results done on multiple datasets establish the efficacy of our method compared to the SOA.

The main limitation of this work is the inclusion of proper texture information in the generated 3D scene and its objects. This module can be stitched into the existing pipeline easily in the future.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The authors did not receive support from any organization for the submitted work. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Abdulla W (2017) Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/mask_RCNN
2. Agarwal S, Snavely N, Seitz SM, Szeliski R (2010) Bundle adjustment in the large. In: Daniilidis K, Maragos P, Paragios N (eds) Computer vision – ECCV 2010, Springer Berlin Heidelberg, pp 29–42
3. Amenta N, Attali D, Devillers O (2020) Complexity of delaunay triangulation for points on lower-dimensional polyhedra. https://doi.org/10.1145/1283383.1283502
4. Apt0 Dataset (2022). http://graphics.stanford.edu/projects/bundlefusion/data/apt0/apt0.zip

5. Arie-Nachimson M, Kovalsky SZ, Kemelmacher-Shlizerman I, Singer A, Basri R (2012) Global motion estimation from point matches. In: 2012 Second international conference on 3D imaging, modeling, processing, visualization & transmission, pp 81–88. https://doi.org/10.1109/3DIMPVT.2012.46
6. Azinović D, Martin-Brualla R, Goldman DB, Nießner M, Thies J (2022) Neural rgb-d surface reconstruction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6290–6301
7. Barua HB, Sarkar C, Kumar AA, Pal A et al (2020) I can attend a meeting too! towards a human-like telepresence avatar robot to attend meeting on your behalf. arXiv:2006.15647
8. Bôcher M (1915) Plane analytic geometry: with introductory chapters on the differential calculus, H. Holt. https://books.google.co.in/books?id=bYkLAAAAYAAJ
9. Borouchaki H, George PL, Hecht F, Laug P, Saltel E (1997) De launay mesh generation governed by metric specifications. part i. algorithms. Finite Elements Anal Design 25(1):61–83. Adaptive meshing, Part 1. https://doi.org/10.1016/S0168-874X(96)00057-1, http://www.sciencedirect.com/science/article/pii/S0168874X96000571
10. Bozic A, Zollhöfer M, Theobalt C, Nießner M (2020) Deepdeform: learning non-rigid RGB-D reconstruction with semi-supervised data. CoRR arXiv:1912.04302
11. Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H et al (2015) Shapenet: an information-rich 3d model repository. arXiv:1512.03012
12. Concha A, Civera J (2015) Dpptam: dense piecewise planar tracking and mapping from a monocular sequence. https://doi.org/10.1109/IROS.2015.7354184
13. Copyroom Dataset (2022). http://graphics.stanford.edu/projects/bundlefusion/data/copyroom/copyroom.zip
14. Cui Z, Tan P (2015) Global structure-from-motion by similarity averaging. In: 2015 IEEE international conference on computer vision (ICCV), pp 864–872. https://doi.org/10.1109/ICCV.2015.105
15. Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) Scannet: richly-annotated 3d reconstructions of indoor scenes. In: Proc computer vision and pattern recognition (CVPR). IEEE
16. Dai A, Nießner M, Zollhöfer M, Izadi S, Theobalt C (2017) Bundlefusion: real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Trans Graph (ToG) 36(4):1
17. Deng K, Liu A, Zhu J, Ramanan D (2020) Depth-supervised nerf: fewer views and faster training for free. CoRR arXiv:2107.02791
18. Deng K, Liu A, Zhu J-Y, Ramanan D (2022) Depth-supervised neRF: fewer views and faster training for free. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
19. Dong W, Shi J, Tang W, Wang X, Zha H (2018) An efficient volumetric mesh representation for real-time scene reconstruction using spatial hashing. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6323–6330
20. Double2 Robotics (2022). https://www.doublerobotics.com/double2.html
21. Fadzli FE, Ismail AW, Aladin MYF, Othman NZS (2020) A review of mixed reality telepresence. In: IOP conference series: materials science and engineering, vol 864, IOP publishing, p 012081
22. Fan H, Su H, Guibas LJ (2017) A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 605–613
23. Farenzena M, Fusiello A, Gherardi R (2009) Structure-and-motion pipeline on a hierarchical cluster tree. In: 2009 IEEE 12th international conference on computer vision workshops, ICCV workshops, pp 1489–1496. https://doi.org/10.1109/ICCVW.2009.5457435
24. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395. https://doi.org/10.1145/358669.358692
25. Forster C, Carlone L, Dellaert F, Scaramuzza D (2015) Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. https://doi.org/10.15607/RSS.2015.XI.006
26. Fu K, Peng J, He Q, Zhang H (2021) Single image 3d object reconstruction based on deep learning: a review. Multimed Tools Appl 80(1):463–498. https://doi.org/10.1007/s11042-020-09722-8
27. Furukawa Y, Ponce J (2009) Accurate, dense, and robust multiview stereopsis. IEEE Trans Pattern Anal Mach Intell 32(8):1362–1376
28. Geneva P, Eckenhoff K, Yang Y, Huang G (2018) Lips: Lidar-inertial 3d plane slam. https://doi.org/10.1109/IROS.2018.8594463
29. Grant WS, Voorhies RC, Itti L (2019) Efficient velodyne slam with point and plane features. Auton Robot 43(5):1207–1224
30. Greene W, Roy N (2017) Flame: fast lightweight mesh estimation using variational smoothing on delaunay graphs, pp 4696–4704. https://doi.org/10.1109/ICCV.2017.502

31. Guo H, Peng S, Lin H, Wang Q, Zhang G, Bao H, Zhou X (2022) Neural 3d scene reconstruction with the manhattan-world assumption. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5511–5520
32. Guo H-J, Prabhakaran B (2022) Hololens 2 technical evaluation as mixed reality guide, https://doi.org/10.48550. arXiv:2207.09554
33. Han X-F, Laga H, Bennamoun M (2019) Image-based 3d object reconstruction: state-of-the-art and trends in the deep learning era. IEEE Trans Pattern Anal Mach Intell 43(5):1578–1604
34. Han X-F, Laga H, Bennamoun M (2021) Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. IEEE Trans Pattern Anal Mach Intell 43(5):1578–1604. https://doi.org/10.1109/TPAMI.2019.2954885
35. Hartley R, Zisserman A (2004) 3D reconstruction of cameras and structure, 2nd edn. Cambridge University Press, pp 262–278. https://doi.org/10.1017/CBO9780511811685.015
36. Haykin S (2009) Neural networks: a comprehensive foundation, 2nd edn. The name of the publisher
37. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: 2017 IEEE international conference on computer vision (ICCV), pp 2980–2988. https://doi.org/10.1109/ICCV.2017.322
38. Henderson P, Tsiminaki V, Lampert CH (2020) Leveraging 2d data to learn textured 3d mesh generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7498–7507
39. Hosseinzadeh M, Latif Y, Pham T, Suenderhauf N, Reid I (2018) Structure aware slam using quadrics and planes. In: Asian conference on computer vision. Springer, pp 410–426
40. Huang J, Dai A, Guibas L, Nießner M (2017) 3dlite: towards commodity 3d scanning for content creation. ACM Trans Graph (TOG)
41. Igarashi T, Moscovich T, Hughes JF (2005) As-rigid-as-possible shape manipulation. ACM Trans Graph (TOG) 24(3):1134–1141
42. Junior ADMB, Neto AADD, De Melo JD (2022) A self-organized neural network for 3d surface reconstruction. ResearchGate
43. Kaess M (2015) Simultaneous localization and mapping with infinite planes. Proc IEEE Int Conf Robot Autom (ICRA) 2015:4605–4611. https://doi.org/10.1109/ICRA.2015.7139837
44. Kang Z, Yang J, Yang Z, Cheng S (2020) A review of techniques for 3d reconstruction of indoor environments. ISPRS Int J Geo-Inf 9(5):330
45. Kean S, Hall JC, Perry P (2011) Microsoft's Kinect SDK, Apress, Berkeley, CA, pp 151–173. https://doi.org/10.1007/978-1-4302-3889-8_8
46. Keselman L, Woodfill JI, Grunnet-Jepsen A, Bhowmik A (2020) Intel realsense stereoscopic depth cameras. CoRR arXiv:1705.05548
47. Laidlow T, Czarnowski J, Leutenegger S (2019) Deepfusion: real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions. In: 2019 International conference on robotics and automation (ICRA). IEEE, pp 4068–4074
48. Lee W, Hasan S, Shamsuddin S, Lopes N (2017) Gpumlib: deep learning som library for surface reconstruction
49. Liu C, Kim K, Gu J, Furukawa Y, Kautz J (2019) Planercnn: 3d plane detection and reconstruction from a single image. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
50. Liu C, Yang J, Ceylan D, Yumer E, Furukawa Y (2018) Planenet: piece-wise planar reconstruction from a single rgb image. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
51. Maity S, Saha A, Bhowmick B (2017) Edge slam: edge points based monocular visual slam, pp 2408–2417. https://doi.org/10.1109/ICCVW.2017.284
52. Marion P, Florence PR, Manuelli L, Tedrake R (2018) Label fusion: a pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1–8
53. McMullen P (1970) The maximum numbers of faces of a convex polytope. Mathematika 17(2):179–184. https://doi.org/10.1112/S0025579300002850
54. Mei C, Benhimane S, Malis E, Rives P (2020) Homography-based tracking and 3d reconstruction for single viewpoint sensors
55. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R (2020) Nerf: representing scenes as neural radiance fields for view synthesis. CoRR arXiv:2003.08934
56. Models FD (2022) Free3d Dataset. https://free3d.com/3d-models/
57. Models CD (2022) CadNav 3D Dataset. https://www.cadnav.com/3d-models/sort-17-2.html
58. Mourikis A, Roumeliotis S (2007) A multi-state constraint kalman filter for vision-aided inertial navigation, vol 22, pp 3565–3572. https://doi.org/10.1109/ROBOT.2007.364024

59. Mur-Artal R, Tardos J (2020) Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. IEEE Trans Robot PP. https://doi.org/10.1109/TRO.2017.2705103
60. Nechaeva O (2005) Neural network approach for parallel construction of adaptive meshes. In: International conference on parallel computing technologies. Springer, pp 446–451
61. Office3 Dataset (2022). http://graphics.stanford.edu/projects/bundlefusion/data/office3/office3.zip
62. Ouyang Z, Liu Y, Zhang C, Niu J (2017) A cgans-based scene reconstruction model using lidar point cloud. In: 2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC), pp 1107–1114. https://doi.org/10.1109/ISPA/IUCC.2017.00167
63. Pan Y, Han Y, Wang L, Chen J, Meng H, Wang G, Zhang Z, Wang S (2019) 3d Reconstruction of ground crops based on airborne lidar technology. IFAC-PapersOnLine 52(24):35–40. 5th IFAC symposium on telematics applications TA 2019. https://doi.org/10.1016/j.ifacol.2019.12.376. https://www.sciencedirect.com/science/article/pii/S2405896319322761
64. RealSense (2022). https://www.intel.in/content/www/in/en/architecture-and-technology/realsense-overview.html
65. Roldão L, De Charette R, Verroust-Blondet A (2020) 3d Surface reconstruction from voxel-based lidar data. CoRR arXiv:1906.10515
66. Rosinol A, Sattler T, Pollefeys M, Carlone L (2019) Incremental visual-inertial 3d mesh generation with structural regularities, pp 8220–8226. https://doi.org/10.1109/ICRA.2019.8794456
67. Salas-Moreno R, Glocken B, Kelly P, Davison A (2014) Dense planar slam, pp 157–164. https://doi.org/10.1109/ISMAR.2014.6948422
68. Sau A, Roychoudhury RD, Barua HB, Sarkar C, Paul S, Bhowmick B, Pal A et al (2020) Edge-centric telepresence avatar robot for geographically distributed environment. arXiv:2007.12990
69. Schreiberhuber S, Prankl J, Patten T, Vincze M (2019) Scalablefusion: high-resolution mesh-based real-time 3d reconstruction
70. Steffen L, Ulbrich S, Roennau A, Dillmann R (2019) Multi-view 3d reconstruction with self-organizing maps on event-based data. In: 2019 19th International conference on advanced robotics (ICAR). IEEE, pp 501–508
71. Stotko P, Krumpen S, Hullin M, Weinmann M, Klein R (2019) Slamcast: large-scale, real-time 3d reconstruction and streaming for immersive multi-client live telepresence. IEEE Trans Vis Comput Graph 25:2102–2112. https://doi.org/10.1109/TVCG.2019.2899231
72. Sun Q, Yuan J, Zhang X, Sun F (2017) Rgb-d slam in indoor environments with sting-based plane feature extraction. IEEE/ASME Trans Mechatronics PP:1–1. https://doi.org/10.1109/TMECH.2017.2773576
73. Tachella J, Altmann Y, Mellado N, McCarthy A, Tobin R, Buller G, Tourneret J-Y, Stephen M (2020) Real-time 3d reconstruction from single-photon lidar data using plug-and-play point cloud denoisers. Nature Commun, vol 10. https://doi.org/10.1038/s41467-019-12943-7
74. Taguchi Y, Jian Y-D, Ramalingam S, Feng C (2013) Point-plane slam for hand-held 3d sensors. In: 2013 IEEE international conference on robotics and automation. IEEE, pp 5182–5189
75. Toldo R, Gherardi R, Farenzena M, Fusiello A (2015) Hierarchical structure-and-motion recovery from uncalibrated images. Comput Vis Image Understand 140:127–143. https://doi.org/10.1016/j.cviu.2015.05.011 . https://www.sciencedirect.com/science/article/pii/S1077314215001228
76. Tychola KA, Tsimperidis I, Papakostas GA (2022) On 3d reconstruction using rgb-d cameras. Digital 2(3):401–421. https://doi.org/10.3390/digital2030022. https://www.mdpi.com/2673-6470/2/3/22
77. Wang C, Guo X (2020) Efficient plane-based optimization of geometry and texture for indoor RGB-D reconstruction. CoRR arXiv:1905.08853
78. Wang J, Song J, Zhao L, Huang S, Xiong R (2019) A submap joining algorithm for 3d reconstruction using an rgb-d camera based on point and plane features. Robot Auton Syst 118:93–111
79. Xie Y, Shu F, Rambach JR, Pagani A, Stricker D (2021) Planerecnet: multi-task learning with cross-task consistency for piece-wise plane detection and reconstruction from a single rgb image. In: British machine vision conference
80. Yang Y, Geneva P, Zuo X, Eckenhoff K, Liu Y, Huang G (2019) Tightly-coupled aided inertial navigation with point and plane features. In: 2019 International conference on robotics and automation (ICRA). IEEE, pp 6094–6100
81. Yang F, Zhou Z (2018) Recovering 3d planes from a single image via convolutional neural networks. In: ECCV
82. Yoon M, Ivrissimtzis IP, Lee S (2008) Self-organising maps for implicit surface reconstruction. In: TPCG, pp 83–90
83. Yu Z, Peng S, Niemeyer M, Sattler T, Geiger A (2022) Monosdf: exploring monocular geometric cues for neural implicit surface reconstruction. Adv Neural Inf Process Syst (NeurIPS)

84. Zhou L, Koppel D, Ju H, Steinbruecker F, Kaess M (2020) An efficient planar bundle adjustment algorithm. In: 2020 IEEE international symposium on mixed and augmented reality (ISMAR). IEEE, pp 136–145
85. Zhu Z, Peng S, Larsson V, Xu W, Bao H, Cui Z, Oswald MR, Pollefeys M (2022) Nice-slam: neural implicit scalable encoding for slam. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
86. cgtrader (2022). Free 3D models. https://www.cgtrader.com/free-3d-models?file_types%5B%%5D=12&page=6&polygons=lt_5k
87. Özyesil O, Voroninski V, Basri R, Singer A (2020) A survey on structure from motion. CoRR arXiv:1701.08493