ObjectFusion: Accurate Object-level SLAM with Neural Object Priors

Zi-Xin Zou
Tsinghua University
zouzx19@mails.tsinghua.edu.cn

Shi-Sheng Huang
Beijing Normal University
huangss@bnu.edu.cn

Tai-Jiang Mu
Tsinghua University
taijiang@tsinghua.edu.cn

Yu-Ping Wang
Tsinghua University
wyp@mail.tsinghua.edu.cn

Abstract

Previous object-level Simultaneous Localization and Mapping (SLAM) approaches still fail to create high quality object-oriented 3D map in an efficient way. The main challenges come from how to represent the object shape *effectively* and how to apply such object representation to accurate *online* camera tracking *efficiently*. In this paper, we provide *ObjectFusion* as a novel *object*-level SLAM which efficiently creates object-oriented 3D map with high-quality object reconstruction, by leveraging neural object priors. We propose a neural object representation with only a single encoder-decoder network to effectively express the object shape across various categories, which benefits high quality reconstruction of object instance. More importantly, we propose to *convert* such neural object representation as precise measurements to jointly optimize the *object shape*, *object pose* and *camera pose* for the final accurate 3D object reconstruction. With extensive evaluations on synthetic and real-world RGB-D datasets, we show that our ObjectFusion outperforms previous approaches, with better object reconstruction quality, using much less memory footprint, and in a more efficient way, especially at the *object* level.

1. Introduction

Object-oriented 3D map often serves as the basis for indoor scene understanding and 3D mapping [26, 31, 49, 52], which can facilitate many applications like intelligent robots, autonomous driving, virtual/augmented reality, etc. Accurate and efficient object-oriented 3D map generation, especially the object-level Simultaneous Localization and Mapping (SLAM) [28, 43, 45, 46, 51], has been receiving continuous research interest from computer graphics and computer vision communities these years. However, previous object-level SLAM approaches would often lead to incomplete object reconstruction [28] when unsatisfactorily scanned by non-professional consumers, or fail to recover 3D shapes of various categories [43, 51] in an efficient and accurate way [45].

One basic challenge to recover high-quality object reconstruction comes from the shape's geometry representation. The implicit function on volumetric voxels, such as signed distance function (SDF) [8], has the advantage of representing 3D objects or scenes at any topology, thus serving as the fundamental shape geometry representation for current mainstream online 3D reconstruction approaches [3, 10, 35, 36], object-level SLAM both in static [28, 51] and dynamic [41] scenarios. Nevertheless, this volumetric representation often needs to allocate a huge amount of GPU memories even for tiny objects, leading to heavy object fusion systems [28]. Besides, holes or large missing object surface regions could not be easily *completed* since no extra geometry priors could be provided by such representation.

The latest neural implicit function [4, 22, 37] and its success for online 3D scene reconstruction (DI-Fusion [19], iMAP [45]), motivate us to introduce neural implicit function as a basic representation for object-level SLAM. NodeSLAM [46] provides probably the first object-level SLAM system which leverages neural implicit function as object representation. However, when applying such neural implicit function for object reconstruction in object-level SLAM, the *object shape*, *object pose* and *camera pose* are often tightly entangled [42] within the neural implicit function, making it difficult to *decode* the geometry priors as direct cues for accurate camera tracking in an efficient way.So it still remains to be a challenging problem to leverage the neural object priors to object-level SLAM, i.e., efficiently employing the direct object priors as accurate cues without sacrificing its effectiveness, to achieve high-quality object-orientated 3D map generation.

In this paper, we propose to express object shape as a novel deep implicit object representation. Unlike the occupancy probability used in NodeSLAM [46], our representation encodes the object shape as an implicit function using

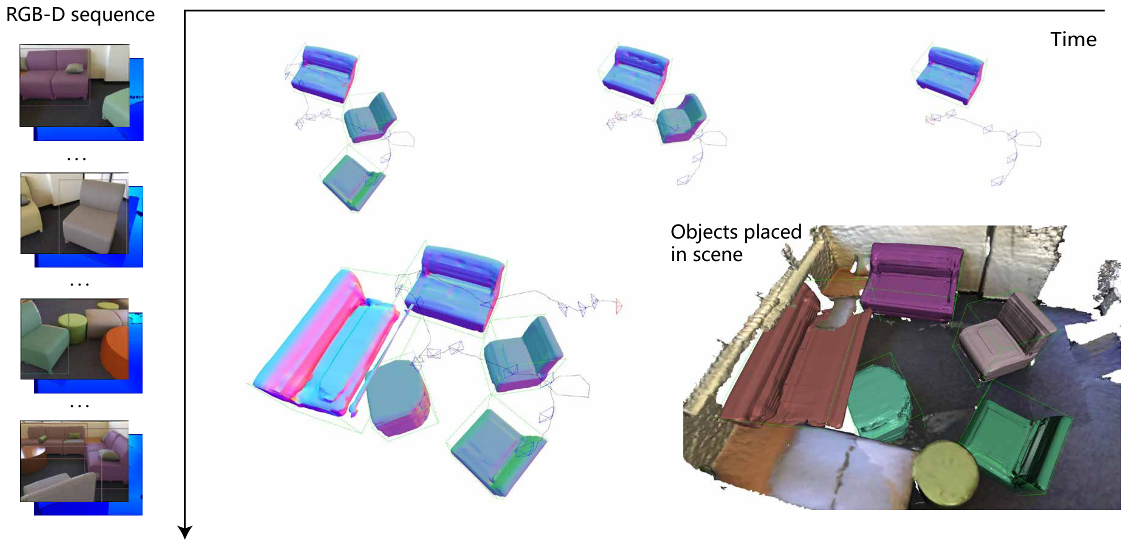

Figure 1. An example demonstration of our ObjectFusion evaluated on ScanNet dataset. Our approach incrementally builds up a map of objects represented using deep implicit object representation from an RGB-D sequence. For real-time visualization, object meshes are extracted at lower (32^3) resolution. The final object meshes which are extracted at higher (256^3) resolution and placed in background scene are shown at right-bottom. Please refer to our supplementary video for more results.

a simple MLP-based encoder-decoder network, which can effectively express the shape details across various object categories. What's more, to convert the object representation as cues for camera tracking and object reconstruction, we propose to directly decode the deep implicit object representation as precise measurements efficiently, without using the time-consuming neural rendering modules as in [45, 46]. Specifically, we propose an accurate camera pose estimation directly based on the deep implicit object representation using a hybrid frame-to-model camera tracking. Furthermore, we formulate the deep implicit object representation as several effective error terms in a joint optimization to further refine the *object shape*, *object pose* and *camera pose* within a local sliding window, thus achieving the final object-orientated 3D map with high quality in an efficient way.

To validate the effectiveness of our approach, we conduct extensive experiments on both synthetic and real-world RGB-D public datasets, such as SceneNet RGB-D dataset [29] and ScanNet dataset [9]. The results show that our approach can produce better object reconstruction quality with *completed* and *detailed* shape than previous TSDF-based object-level SLAM approaches [28, 51], while maintaining at least comparable (sometimes better) camera pose estimation accuracy, thus serving as a new state-of-the-art object-level SLAM method for accurate and efficient object-orientated 3D map generation.

2. Related Work

Our work aims at an accurate object-orientated 3D map generation using a visual SLAM based on RGB-D. The visual SLAM is a continuous popular SLAM technique with many exciting works such as MonoSLAM [11], LSD-SLAM [14], ORB-SLAM2 [34], RKSLAM [25] and DSO [13] for static scenes, PL-SLAM [16] for structured scenes (with point and line features [20]) and LCCRF-SLAM [12] for dynamic scenes. Here we only discuss relevant works including online 3D reconstruction, deep 3D representation and reconstruction and object-level SLAM, and refer readers to [1] for a dedicated survey on the progress of visual SLAM in the past few decades.

2.1. Online 3D Reconstruction

Inspired by the pioneering work of KinectFusion [48], the online 3D reconstruction has achieved much progress in the past decades. VoxelHash [36] and its variations [6] provided an efficient sparse voxel allocation mechanism, which is capable of reconstructing large scale 3D scenes efficiently. The subsequent approaches introduce techniques such as global pose graph (e.g., InfiniTAM [23]), bundle adjustment (e.g., BundleFusion [10], SemanticFusion [21], Noise Resilient Fusion [3]), or deformable loop closure [48] to reconstruct globally consistent 3D scenes.

One of the main drawbacks of the previous online 3D reconstruction approaches is that they rely on memory-consuming geometry representation, i.e., the signed distance function (SDF) on volumetric voxels [8], which often leads to a huge amount of GPU memory consumption. Different from those previous approaches, we introduce a novel deep implicit object representation to express the object shape. Based on such representation, we provide an accurate object-level SLAM approach with much less memory footprint, while achieving *completed* and *detailed* 3D object reconstruction of high quality.

2.2. Deep 3D Representation and Reconstruction

With the huge progress of deep geometry learning [50], many deep 3D representations for objects or scenes have been proposed these years. DeepSDF [38] probably for the first time proposes to formulate the implicit function as an MLP-based deep neural network, i.e., neural implicit function, which enables effective single-view 3D reconstruction and shape interpolation. DeepLS [4] and LIG [22] propose to express the local shapes as a neural implicit function, thus being able to effectively represent complex scenes or objects across various categories. Convolutional Occupancy Network [39] introduces a more flexible neural implicit representation by combining the convolutional encoders and implicit occupancy decoders, which shows the impressive ability for high-fidelity 3D reconstruction. Based on such neural implicit function, many 3D reconstruction approaches have been proposed for high-quality 3D object or scene reconstruction. DI-Fusion[19] is one of the first approaches to leverage a deep 3D representation for online 3D reconstruction, and achieves impressive 3D reconstruction results. iMAP [45] adopts to use NeRF [30] as the scene representation, which achieves impressive 3D scene reconstruction results. Given camera poses estimation, FroDo [42] and NeuralRecon [47] can generate 3D object or scene reconstruction from monocular RGB frames with impressively high surface reconstruction quality.

Our approach also adopts to represent the object shape as a neural implicit function. But different from these previous works, we contribute to the way that converts the neural object representation as precise measurements, for accurate camera pose estimation and object surface generation in an efficient way, which aims at high-quality object-oriented 3D map generation.

2.3. Object-level SLAM

As one kind of visual SLAM technique, object-level SLAM adopts to utilize object instances as landmarks for accurate camera tracking and surface mapping. SLAM++[43] for the first time proposes to utilize object priors to detect object landmarks for camera tracking, although the object priors are simply obtained by retrieving in manually collected 3D shape sets. The following works such as Fusion++ [28] directly use the object mask predicted from 2D CNNs to build object landmarks for camera poses estimation, and formulate the object level bundle adjustment to further rectify the global pose estimation. MaskFusion [41], RigidFusion [49], MID-Fusion [51] and EM-Fusion [44] provide accurate object segmentation in dynamic scenes and use the object instances as landmarks for accurate camera tracking or scene reconstruction.

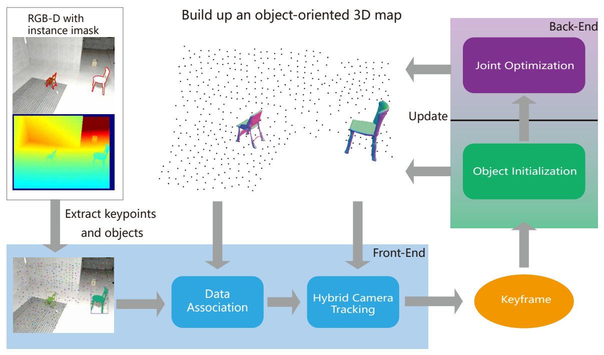

Figure 2. Overview of our ObjectFusion based on deep implicit object representation. ObjectFusion estimates the camera pose of each frame and incrementally builds up 3D surface reconstruction of object instances in the scene.

Unlike these previous object-level SLAM approaches that utilize implicit function as object representation, our approach proposes to express the object shape as deep implicit object representation, which can effectively represent the fine geometry details even across various object categories. Benefiting from the deep implicit object representation, we show that our approach can achieve much better object reconstruction quality in terms of *completion* and *details* than the previous approaches, and serves as a new state-of-the-art object-level SLAM approach.

3. ObjectFusion

Overview. The overview of our system is illustrated in Fig. 2. Given a stream of RGB-D images, our ObjectFusion system estimates the camera pose of each frame and incrementally builds up 3D surface reconstruction of object instances in the scene, leading to an object-oriented 3D map as output. Specifically, for each coming RGB-D frame, we first detect the instance segmentation masks in the frame, and then encode each object instance to a latent vector using a deep implicit object representation (see Sec. 3.1). Different from traditional object representations such as volumetric voxels or discretized surfels, our representation can effectively learn the object geometry priors, and can reconstruct the final mesh at arbitrary resolution and topology. For each detected object instance, we perform object-level data association and initialize the object shape and pose (see Sec. 3.3). Then the camera pose is estimated using our hybrid camera tracking which is based both on the deep implicit object representation and sparsely sampled map points (see Sec. 3.4). Finally, in order to obtain the globally consistent object shape and pose, we build up a joint optimization for *object shape*, *object pose*, and *camera pose* in a sliding keyframe window (see Sec. 3.5).

Data Association. After extracting objects from RGB image with instance masks, we need to associate them with existing objects in the map. To this end, we project the 3D bounding boxes of all existing objects in the map to the current frame, and match these projected 2D bounding boxes with the mask bounding boxes by computing their Intersection over Union (IoU). Kuhn-Munkres Algorithm [32] is applied to solve this linear assignment problem. If an object instance is not associated with any existing object, we initialize a new object in the map.

3.1. Deep Implicit Object Representation

Different from the implicit function representation used in previous object-level SLAM approaches, we propose to represent the object shape using the deep implicit object representation. Unlike NodeSLAM[46] encoding the object shape with voxel occupancy grids which has limited representation ability for complex objects, we adopt the implicit function, i.e., SDF, to represent the object underlying 3D surface, and formulate it as a deep neural network following the DeepSDF architecture[37], leading to a deep implicit object representation. Specifically, we adopt an encoder-decoder architecture for the deep implicit representation, in which the predicted object instance is encoded as a latent vector and then decoded as the SDF of the object shape. The structure of our deep implicit object representation is illustrated in Fig. 3.

In the deep implicit object representation, we use an L-dimension latent vector l for the object shape and use T_{ow} to represent the object pose. T_{ow} denotes a transformation from the world coordinate system to object canonical coordinate system. For decoder ϕ_d, we compute the signed distance s at any position $p \in R^3$ concatenated with the latent vector l as $s = \phi_d(p, l)$. The final underlying surface is extracted as the zero iso-surface using Marching Cubes [27].

Following the automatic derivative mechanism from the deep neural network, we can easily calculate the spatial derivative of $\frac{\partial \phi_d(p,l)}{\partial p}$ via back-propagation of the deep implicit object representation, which can be further used in the camera tracking and joint optimization described later.

3.2. Object Shape and Pose Inference

One important point for our deep implicit object representation is how to infer the object *shape* and *pose* from such neural representation accurately and efficiently. Unlike NodeSLAM[46] and iMAP[45] that infer the object shape and pose via time consuming neural rendering modules, we propose to address this issue in an optimization way. Given initial object shape (encoded in latent vector) and object pose, our strategy is to iteratively update the object latent vector and pose using an object shape and pose optimization. For inference accuracy, we introduce hybrid cues and formulate them as precise measurements into the optimization. For efficiency, we adopt to optimize the object shape and pose in an alternately iterative way, i.e., we first optimize the object shape with a *shape inference* by fixing the object pose and then optimize the object pose with a *pose inference* by fixing the object shape. In this way, we perform inference of object shape and pose accurately and efficiently from our deep implicit object representation.

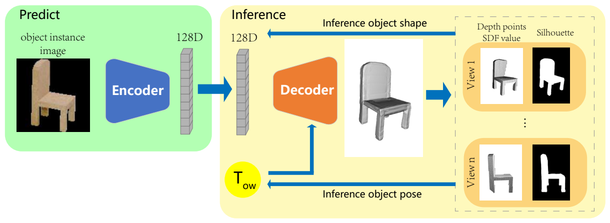

Figure 3. The backbone of our deep implicit object representation. The encoder encodes an object instance image as a latent vector, and then is decoded as a signed distance function of the object. The signed distance value of surface points (depth) and projection silhouette are used object shape and pose inference.

Shape Inference. Given the initial object latent vector l_0 from the deep implicit object representation, and transformation T_{oc} from RGB-D frame to object canonical coordinate, which can be calculated by given camera pose T_{wc} as $T_{oc} = T_{ow} T_{wc}$, we seek to find the optimized latent vector l^* by minimizing the object function E_{shape} considering the geometry term E_g, silhouette term E_s, and regularization term E_r as:

$$l^* = \arg\min_{l} \{E_{shape} = E_g + w_1 \cdot E_s + w_2 \cdot E_r\} \tag{1}$$

where w_1 and w_2 are weight parameters that balance the silhouette term and regularization term respectively.

The geometry term E_g measures the SDF errors from each depth point $p \in P$ of the current RGB-D frame, by considering both the SDF value from the point p and its normal information n respectively as:

$$\begin{aligned} E_g(T_{oc}, l) = \frac{1}{2|P|} \sum_{p \in P} \{ & L(\phi_d(T_{oc} p, l), 0) \\ & + L(\phi_d(T_{oc} p + (T_{oc}^{-\mathbf{T}} n) \cdot t, l), t)\} \end{aligned} \tag{2}$$

$$L(a, b) = |clamp(a, \delta) - clamp(b, \delta)| \tag{3}$$

$L(a, b)$ is the clipped L1 loss function with threshold $\delta = 0.1$, and t denotes the step length parameter.

For the silhouette term E_s, we follow the mechanism introduced by [40, 42] which casts rays from the object's mask, uniformly samples points along the rays, and calculates the probability hitting function from the object surface as:

$$\begin{aligned} E_s(T_{oc}, l) = \int_{\Omega} \{ & H(T_{oc}, l) P_f(x) \\ & + (1 - H(T_{oc}, l)) P_b(x) \} d\Omega \end{aligned} \tag{4}$$

$$H(T_{oc}, l) = 1 - exp(\sum_{p \ on \ ray} log(1 - \frac{e^{\phi_d(T_{oc}p, l)\zeta}}{e^{\phi_d(T_{oc}p, l)\zeta} + 1})) \tag{5}$$

where P_f and P_b are two parameters used to describe the probability for each pixel whether it belongs to object instance or the background region respectively, with $P_f + P_b = 1$. Specifically, we set $P_f = 1$ for pixels x located in object instance mask, otherwise set $P_b = 1$ for those located in background.Parameter ζ controls the function smoothness and is set to 100.

For the regularization term E_r, we adopt to regularize the latent vector l as $E_r = \|l\|^2$.

Pose Inference. Given the optimized latent vector l^* for the object shape, we seek to find an optimized object pose T^*_{ow} for each object instance. Our observation is that by applying the optimized transformation T^*_{ow}, the decoded SDF values for depth point $p \in P$ in the current RGB-D frame from the decoder $\phi_d(p, l^*)$ should reach the minimal value, ideally i.e., zero. So we formulate the pose inference with a least-squares optimization as:

$$T^*_{ow} = \arg \min_{T_{ow}} \{ E_{pose}(T_{ow}) = \sum_{p \in P} \rho(\|\phi_d(T_{ow} T_{wc} p, l^*)\|^2) \} \tag{6}$$

with $\rho(\cdot)$ denoting a Huber robust function.

3.3. Object Shape and Pose Initialization

Our object shape and pose inference needs good initial guesses, which could accelerate the convergence of both shape inference and pose inference for the final accurate object shape and pose.

Shape initialization. DeepSDF [37] proposes to initialize the object latent vector as a prior with a zero-mean multivariate-Gaussian distribution. However, this initialization strategy is only suitable for limited object class. NodeSLAM[46] uses a one-hot vector for different object classes, but the convergence is slow as extra optimization iteration is needed during the optimization.

Inspired by FroDo [42], we adopt to train the encoder network with only object instances, and take the predicted latent vector as the object shape initialization latent vector, which can cover different categories of objects. Specifically, we use ResNet50 [18] as the backbone of the encoder network and modify the output variable dimension as the size of the object's latent vector, i.e., 128.

Pose initialization. A good initialization for object pose is also important for shape inference. Here, we assume that all objects are placed vertically on the scene floor, where the floor plane is estimated by the point cloud of floor sampled in the view of all the past frames. Besides, since the coordinates of ShapeNet[5] models are between [-1, 1], empirically, we resize the scale of the transformed point cloud in object canonical coordinate within the range of [-1, 1]. Besides, we enumerate $K = 15$ rotation angle uniformly sampled within 360° around the vertical axis of the floor, and use them as the initial object pose for the subsequent shape inference.

With the above shape and pose initialization, we perform the object shape and pose inference described in Section 3.2 to generate the object shape and pose respectively, which is further used to track camera poses. Besides, for real time front-end tracking, we perform the object and pose initialization and inference in a parallel thread at back-end.

3.4. Camera Tracking

After building up the object shape and pose inference from the deep implicit object representation, we propose a hybrid camera tracking to estimate the camera pose of each RGB-D frame directly based on the deep implicit object representation. Specifically, at each timestep t, we estimate the camera pose T^t_{wc} in the world coordinate using a set of object instance landmarks $\{O^t_i\}$ and the depth map D^t. First, for each object O^t_i, we project the depth measurement D^t back to 3D point cloud through $P^t_i = \pi^{-1}(M_i(D^t))$ using the given camera intrinsic parameters, where π denotes the projection function and π^{-1} is its reverse function. M_i is the mask function to get the depth measurement of object $\{O^t_i\}$. We make an initialization of the current frame pose T^t_{wc} as its reference keyframe's pose T^r_{wc}. Then our goal is to estimate the camera pose T^t_{wc} by minimizing the following objective function:

$$E(T^t_{wc}) = E_{sp}(T^t_{wc}) + w_3 \cdot E_{obj}(T^t_{wc}), \tag{7}$$

where $E_{sp}(T^t_{wc})$ and $E_{obj}(T^t_{wc})$ are the sparse point term and object term respectively, and parameter w_3 controls the balance between the two terms.

Object Term. Our observation is that the decoded SDF value for points located at the depth measurement of an object should be a small value, ideally i.e. zero, thus the object term is designed to align the underlying 3D surface of the object decoded by the latent vector, which is inspired by SDF tracker[2, 19] that estimates the camera pose by minimizing the signed distance value of objects' depth point cloud P^t_i transformed in the object's canonical coordinate.

We define the object term as:

$$E_{obj}(T_{wc}^t) = \sum_i \sum_{p \in P_i^t} \rho(\left\|\phi_d(T_{ow}^i T_{wc}^t p, l^i)\right\|^2), \tag{8}$$

Since the object term depends on object surface reconstruction in good quality, to make an accurate camera tracking, we only choose 'good' object instances for calculating the object term. Here, we propose to label the object instance as 'good' or 'bad' during the object shape and pose inference (Sec. 3.2). Specifically, if the Chamfer distance between the generated mesh and the corresponding depth measurements is less than the threshold τ_{ok}, we set the object as 'good' to be able to participate in tracking and following pose optimization step. Otherwise, the object is set as 'bad' without being considered as measurement for the object term. And if the Chamfer distance is larger than the threshold τ_{fail}, we will remove them from the map.

Sparse Point Term. Similar to other sparse SLAM systems like ORB-SLAM2 [33], sparse point term is the sum of re-project error between matched 3D map points $X_i \in \mathbb{R}^3$ and 2D keypoints $x_j \in \mathbb{R}^2$, with $(i,j) \in \chi$ the set of all matches:

$$E_{bg}(T_{wc}^t) = \sum_{(i,j) \in \chi} \rho(\|\pi(X_i) - x_j\|^2) \tag{9}$$

where $\rho(\cdot)$ is the Huber robust function. When performing the optimization, we apply Gauss-Newton iterative algorithm to solve the optimization problem efficiently.

3.5. Joint Optimization of Object Shape, Object Pose and Camera Pose

During the long-term camera tracking, the camera poses estimated by our hybrid camera tracking would occur drift. To rectify the motion drift, we also build a joint optimization in the back-end that jointly rectifies the object shape, object pose and camera pose. However, performing such joint optimization for every RGB-D frame would be time-consuming. For time efficiency, we only perform the joint optimization for object shape, object pose and camera pose in the keyframes of a sliding window.

Keyframe Selection. Our keyframe selection strategy consists of four parts. A new keyframe will be inserted when any of the following conditions is met:

1. The number of tracked points in current frame is less than a minimal value $\tau_t = 150$.
2. No new keyframe is inserted for a long time, i.e., the number of frames passed since the last keyframe surpasses a given threshold $\tau_f = 25$.
3. A new object is detected and the number of frames passed since the last keyframe surpasses $\tau_o = 10$, so a keyframe is needed to initialize and create the new object.
4. The frame viewpoint for any of the existing objects is larger than $18°$.

Joint Optimization. For a sliding window of keyframe set $\mathcal{C}$ which contains object instances $\mathcal{O}$, our goal is to jointly optimize the camera pose $T_{wc}^i, i \in \mathcal{C}$, object pose $T_{ow}^j, j \in \mathcal{O}$ and object shape in latent vector l^j. In our experiments, we set the keyframe number of the sliding window as $N = 5$ in the synthetic dataset and $N = 10$ in the real-world dataset.

For efficiency, our joint optimization takes an alternatively iterative strategy to optimize shape and pose respectively, i.e. we firstly apply a *pose optimization step* by fixing all object shapes and then apply a *shape optimization step* by fixing all camera poses and object poses.

Pose Optimization Step. In pose optimization step, our purpose is to minimize the following objective function with respect to sparse point set $\mathcal{P}$ and 'good' object set $\mathcal{O}' \subseteq \mathcal{O}$:

$$\min \sum_{i \in \mathcal{C}, j \in \mathcal{O}'} e_{co}(i,j) + \sum_{i \in \mathcal{C}, k \in \mathcal{P}} e_{cp}(i,k) \tag{10}$$

where the camera-to-object error $e_{co}(i,j)$ and camera-to-point error $e_{cp}(i,k)$ defined as:

$$e_{co}(i,j) = \rho(\left\|\phi_d(T_{ow}^j T_{wc}^i P_i^j, l^j)\right\|^2) \tag{11}$$

$$e_{cp}(i,k) = \rho(\left\|\pi(X^k) - x^k\right\|^2) \tag{12}$$

where x^k is the corresponding 2D points of X^k and P_i^j is the depth point cloud of the object j measured in the camera i. This optimization can be performed by Levenberg-Marquardt algorithm using Schur Compliment efficiently.

Shape Optimization Step. Object shape should be optimized to fit multi-view observations of keyframes in sliding window, we accumulate all objective functions across each of object-camera pairs as the whole shape optimization function:

$$\min \sum_{i \in \mathcal{C}, j \in \mathcal{O}} \{E_g(T_{ow}^j T_{wc}^i, l^j) + w_1 \cdot E_s(T_{ow}^j T_{wc}^i, l^j)\} + \sum_{j \in \mathcal{O}} w_2 \cdot E_r(l^j) \tag{13}$$

where E_g, E_s, E_r are geometry term, silhouette term and regularization term and use the same weight parameters w_1, w_2 as Equation (1). Adam algorithm is applied to optimize the latent vector of all objects in a sliding window.

4. Experiments and Analysis

In this section, we first explain the implementation details of our method and then compare our method with others to demonstrate its effectiveness and efficiency.

Table 1. Comparison of ATE on our synthetic four scenes (measured in centimeters). The best numbers are indicated in boldface.

	S1(1)	S1(2)	S1(3)	S1(4)	S1(5)	S2(1)	S2(2)	S2(3)	S2(4)	S2(5)	S3(1)	S3(2)	S3(3)	S3(4)	S3(5)	S4(1)	S4(2)	S4(3)	S4(4)	S4(5)
IM	6.19	48.58	29.80	17.75	10.11	23.64	87.37	6.29	5.16	17.94	25.61	16.71	12.40	9.29	5.11	60.90	10.33	52.82	37.15	13.03
MF	8.17	28.36	36.42	18.14	13.19	31.07	40.52	4.89	8.57	28.50	16.67	12.73	30.55	7.19	5.35	30.45	4.41	26.57	17.14	16.29
MID	6.20	4.94	11.10	3.74	3.92	4.55	5.32	4.09	4.01	2.69	4.18	-	9.74	4.06	2.69	3.77	3.00	-	5.92	8.84
BF	5.51	4.60	10.34	1.72	3.83	2.78	13.54	3.07	3.92	4.68	7.94	1.84	3.43	3.97	3.64	8.47	1.19	11.02	7.57	10.46
Ours	**1.71**	**0.58**	**1.07**	**0.60**	**0.86**	**1.31**	**1.87**	***0.87***	***0.51***	**1.31**	***0.88***	**0.57**	***1.16***	***1.00***	**0.34**	***0.74***	***0.95***	***1.43***	**1.49**	**1.05**
Ours(w/o Obj)	**0.75**	0.53	**0.57**	**0.34**	0.84	**0.88**	1.34	0.89	0.53	1.49	1.35	0.57	1.31	1.07	0.33	0.94	1.22	1.63	**1.05**	**0.70**
ORB	0.86	**0.30**	0.59	0.58	**0.49**	1.14	**0.48**	**0.27**	**0.34**	1.40	**0.61**	**0.54**	**0.82**	**0.40**	**0.29**	**0.45**	**0.42**	**1.29**	1.14	0.73

4.1. System Implementation

When implementing the encoder-decoder network of deep implicit object representation, we use 8 FC layers for the decoder as like DeepSDF architecture[37], and also train the networks using the 3D models from ShapeNet dataset[5]. Different from DeepSDF[37] which trains an individual decoder for each class, we train one decoder to represent different sets of object shapes using all models together. We use the L2 loss like [24, 42] did to train the encoder to measure the distance between the embedding vector of the model and the predicted latent vector of its rendered image. The rendered images are provided by [7].

Parameters. We set the dimensions of the latent vector in our network to be 128. For optimization term E_{shape}, we set silhouette term weight $w_1 = 1$ in a synthetic dataset and $w_1 = 0.1$ in a real-world dataset, regularization term weight $w_2 = 1e^{-4}$, and step length of sample points along normal direction t in is set as 0.05. Threshold parameters of Huber robust function of object's depth point cloud term (in Equation(6),(8),(11)) and sparse map point term (in Equation(9),(12)) are set 0.05 and $\sqrt{5.991}$ respectively. We also set object term weight in camera tracking (in Equation(7)) as $w_3 = 0.2$.

Dataset. We evaluate our system performance both on a synthetic dataset and a real-world dataset. For the synthetic dataset, we create four synthetic indoor scenes with different objects randomly placed on the floor, and five camera trajectories are randomly generated for each scene. The code for the generation and rendering of this dataset is provided by SceneNet RGBD[29]. We only evaluate the performance of object reconstruction on the most common classes in the indoor scenes, i.e., "Chair", "Table" and "Sofa". It's more difficult to reconstruct these objects since they have more complex shapes and poses than cylindrical objects like "cup", "bottle" or "bowl" which most are axial symmetry with simple shapes in NodeSLAM [46]. For real-world dataset, we adopt ScanNet dataset[9] to evaluate the object reconstruction quality of our approach for a real-world dataset.

4.2. Evaluation on Camera Pose Estimation

In this subsection, we make an evaluation of accuracy of the camera pose estimation of our approach on the synthetic dataset. Specifically, we compare our system with five previous different types of baseline systems, 1) sparse SLAM: ORB-SLAM2(ORB) [34]; 2) TSDF-fusion reconstruction system: InfiniTAM(IM) [23], BundleFusion(BF) [10]; 3) object-level SLAM: MID-Fusion(MID) [51] and MaskFusion(MF) [41]. For the accuracy metric, we adopt Root-Mean-Square-Error (RMSE) of Absolute Trajectory Error (ATE) as a metric of the accuracy of camera pose estimation. ORB-SLAM2 is a sparse feature point-based SLAM, which doesn't reconstruct any object surface. Here, we adopt ORB-SLAM2 as the baseline approach for the camera pose estimation. InfiniTAM, BundleFusion and MID-Fusion are TSDF-based 3D surface reconstruction systems, while MaskFusion is a surfel-based 3D surface reconstruction system. InfiniTAM and BundleFusion aim at 3D reconstruction for the whole scene while MID-Fusion and MaskFusion are object-level SLAM for object reconstruction only. Additionally, we also implement a version of our system without using object landmarks (i.e., tracking only with map points) to evaluate the effect of object term in our camera tracking module.

In order to eliminate the impact from the different implementation of different systems on the results, we perform the evaluation for each sequence five times, and calculate the average ATE accuracies by removing the highest and lowest ATE scores. Tab. 1 shows the detailed ATE scores of five random trajectories (the numbers in brackets are their trajectory index) for each of four synthetic scenes. '-' means that the method gets failed, for which we would not evaluate the accuracy for such scenes. ORB-SLAM2 achieves the lowest tracking ATE accuracy score in most sequences of the synthetic dataset benefiting from its sparse

Table 2. Object reconstruction results comparison on Chamfer distance(CD, measured in centimeters) and completeness(COMP, with 5cm threshold). The best numbers are indicated in boldface.

Methods	Chair		Table	
	CD	COMP	CD	COMP
MID-Fusion	10.12	65.80	13.58	60.94
MaskFusion	13.66	47.81	16.05	35.52
Ours(w/o initial Opt)	7.08	80.31	12.22	66.33
Ours(w/o joint Opt)	5.15	85.07	8.21	79.18
Ours	**4.90**	**86.92**	**7.67**	**81.58**

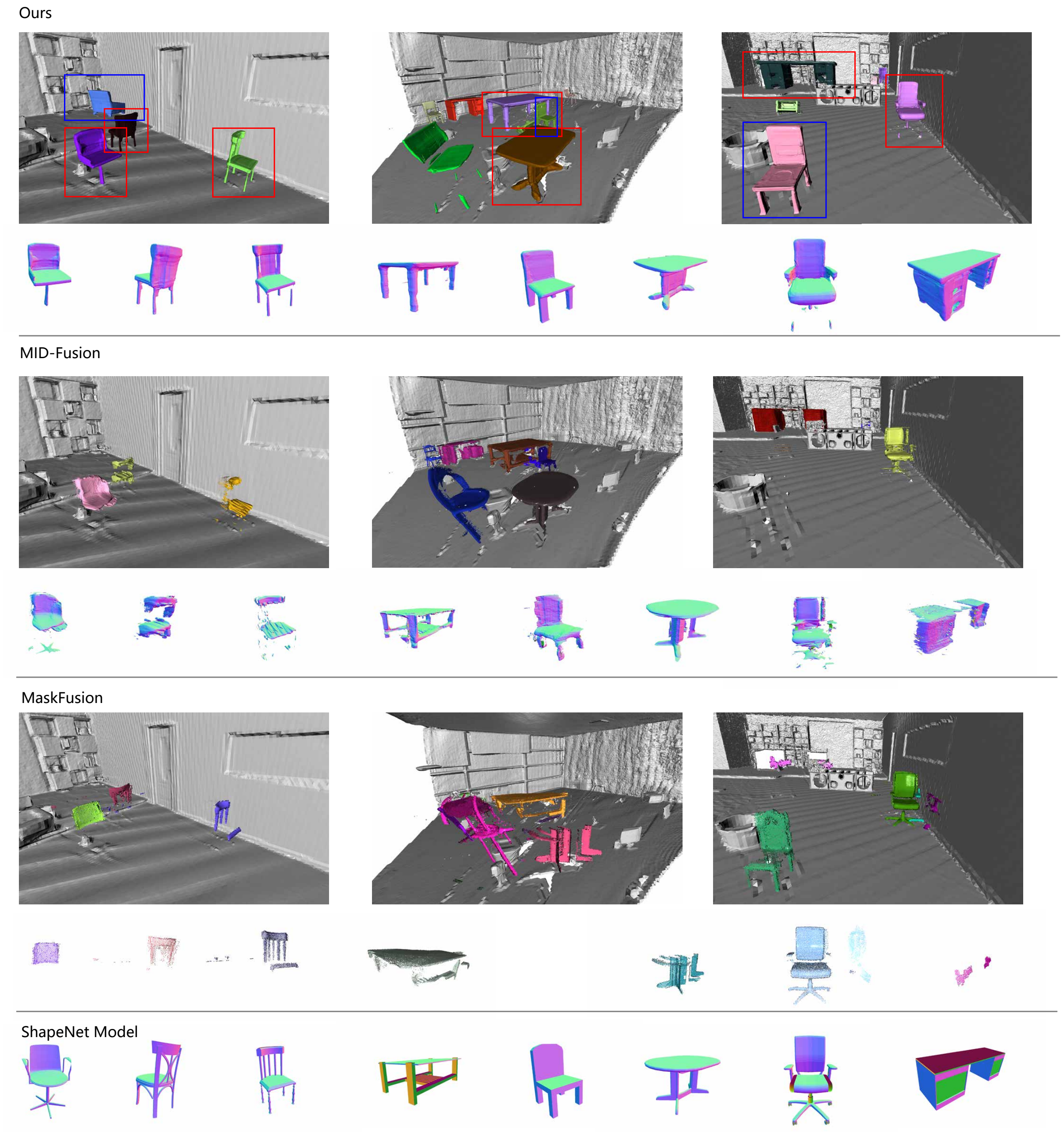

Figure 4. Qualitative scene reconstruction results on our synthetic dataset, with individual object reconstructions (marked as red and blue boxes) are also listed below. Objects marked in blue boxes are failure results from some other method. The ground-truth models from ShapeNet are listed at the bottom row for reference.

feature-point-based tracking method. Our method outperforms InfiniTAM, MID-Fusion, MaskFusion and BundleFusion, which achieves significant ATE score decrement over these four approaches, and obtains a comparable trajectory accuracy level as ORB-SLAM2. For the object term, our approach achieves a slightly better ATE score than the system of Ours (w/o Obj) like some sequences such as S3(1) and S4(2) indicated in italic in Tab. 1.

4.3. Object Reconstruction Quality Evaluation

Our approach can not only accurately estimate the camera pose, but also reconstruct the object's shape in high completeness and quality. In this subsection, we evaluate our system's ability for object reconstruction in

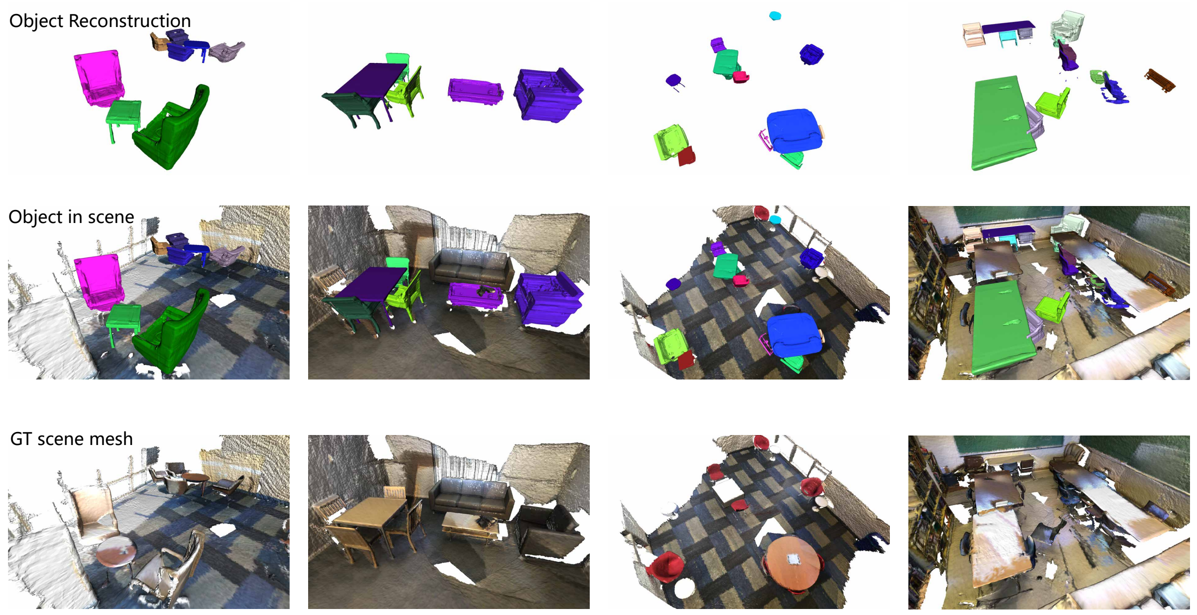

Figure 5. Qualitative scene reconstruction result from ScanNet dataset. Top row: object reconstructed by our approach. Middle row: objects placed in the scenes. Bottom row: ground-truth scene mesh from ScanNet.

terms of both quantitative and qualitative results. We adopt two metrics are used for object shape reconstruction evaluation[42, 46], i.e., Chamfer Distance (CD) and Completeness (COMP), to evaluate the quantitative scores. For Completeness score, we use the threshold as 5cm.

We compare our method against MID-Fusion and MaskFusion, which are two recent object-level SLAM approaches. Although MID-Fusion and MaskFusion focus on object-level SLAM in dynamic scenes, which is different from our goal for static scenes, we perform the comparison on the same dataset of SceneNet RGBD dataset. One of the main drawbacks of MID-Fusion in camera tracking is that its tracking for object instance could be highly unreliable for objects moving out of view frustum, leading to severe object's pose drift and poor results of object reconstruction. Since all sequences of our synthetic dataset are in static environments, we set the object to be static in MID-Fusion to avoid severe object's pose drift. For our approach, we implement two different versions including (1) one without using the object shape and pose inference, termed as Ours (w/o initial Opt), and (2) one without using the joint optimization, termed as Ours (w/o joint Opt).

Tab. 2 lists the object reconstruction results from the five different kinds of approaches. We can see that our approach achieves the lowest surface distance (CD) and highest completeness scores (COMP) comparing the other four approaches. Benefiting from our accurate camera pose estimation and neural object shape representation, our approach can reconstruct more complete objects than MaskFusion and MID-Fusion. Besides, Ours (w/o joint Opt) achieves better CD (5.15) and COMP (8.21) scores than Ours (w/o initial Opt) with CD (7.08) and COMP (12.22), showing that object shape and pose inference plays a major role in object surface reconstruction. Based on this module, the joint optimization further improves the object surface reconstruction quality.

Fig. 4 shows some qualitative results compared with MID-Fusion and MaskFusion, where the corresponding ShapeNet models are shown at the bottom for reference. We can see that our approach achieves more complete and better quality object reconstruction than the traditional TSDF-fusion method (e.g. MID-Fusion) and surfel-fusion method (e.g. MaskFusion). Due to the low accuracy camera pose estimation of MaskFusion which is evaluated in Sec. 4.2, its estimated object pose will drift significantly and often leads to unsatisfied object reconstruction, such as the pink chair in the middle column of MaskFusion in Fig. 4, and incomplete object reconstruction (e.g. chairs in the first column in Fig. 4). Furthermore, our deep implicit representation only represents the object shape using a 128-dimensional vector while MID-Fusion uses a fixed size volumetric voxels that leads to much more memory footprint. For example, we only need 512 KB for encoding one object, which is much less than 64 MB using a 256^3 SDF voxels.

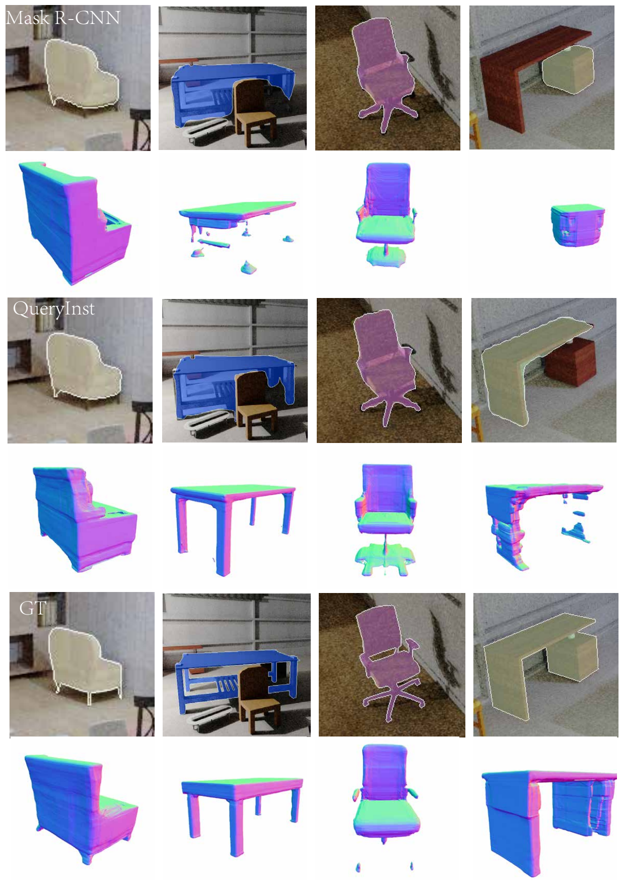

Figure 6. Visual comparison of reconstruction result between Mask R-CNN segmentation, QueryInst segmentation and GroundTruth segmentation.

4.4. Real World Evaluation

We also evaluate qualitative results on ScanNet dataset[9], which contains a large scale of real-world RGB-D sequence and is annotated with instance-level semantic segmentation. Fig. 1 and 5 show some visual effects of the object reconstruction results that our system successfully generates with high-quality reconstruction. The generated object surfaces are complete, sometimes even better than the ground-truth mesh (obtained with 3D reconstruction method [10]) benefit from the neural object priors we used. There would also be some objects that have not been reconstructed by our approach, which is mainly due to occlusion and truncation, being one limitation of our approach.

4.5. Runtime

Our system runs on a platform with an Nvidia RTX 3090 GPU at ∼7 Hz. The average timing of each main component is shown in Tab. 3. "Pre-processing" mainly contains points and objects extraction and data association. "Tracking" represents our proposed hybrid camera tracking. "JO" is the abbreviation of "Joint Optimization". Since the runtime of tracking and joint optimization is related to the number of objects involved, we report average increase in runtime for tracking or joint optimization when the number of involved objects increased by one (marked as '/O') and runtime without any objects respectively. Note that although our deep implicit object representation is more complex than the voxel occupancy grids used in NodeSLAM [46], our approach efficiently converts such deep neural representation to reconstruct heterogeneous object shapes at a fast camera tracking processing rate comparable with NodeSLAM [46].

Table 3. Runtime analysis of our system.

	Pre-processing	Tracking	Tracking/O	JO	JO/O
Time	67ms	20ms	50ms	592ms	887ms

Table 4. Ablation study based on three different kinds of instance segmentation, including Mask R-CNN, QueryInst and GroundTruth.

Methods	Chair		Table	
	CD	COMP	CD	COMP
Ours(MaskRCNN)	6.89	84.19	11.05	67.96
Ours(QueryInst)	6.55	84.23	11.70	67.36
Ours(GroundTruth)	4.90	86.92	7.67	81.58

4.6. System Study

Instance Segmentation. In order to demonstrate the effect of instance segmentation, we make an evaluation between GT and deep models for object instance segmentation. We evaluate two versions of our approach for different deep model segmentation. One uses Mask R-CNN [17] which served as classical segmentation model to generate object instance segmentation, termed as Ours(MaskRCNN). And the other uses QueryInst [15] which served as SOTA segmentation model, termed as Ours(QueryInst). Table. 4 shows the CD and COMP accuracy for the 'Chair' and 'Table' objects using three kinds of instance segmentation respectively. We can see that the accuracy from QueryInst (CD 6.55, COMP 84.23) and MaskRCNN (CD 6.89, COMP 84.19) is slightly worse than GT (CD 4.90, COMP 86.92), which is reasonable since the object segmentation quality would directly influence the shape generation in our approach. However, our approach can still achieve consistent object reconstruction in different quality of object instance segmentation, which demonstrates our approach has good robustness for object generation under different deep models of object instance segmentation. Some visual comparison results are shown in Fig. 6,

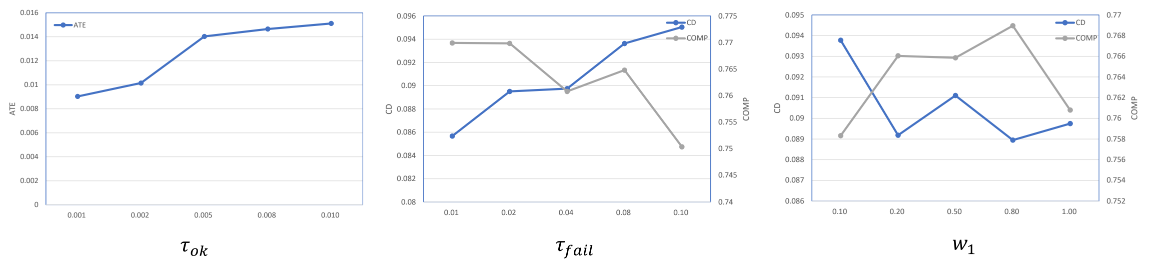

Figure 7. The ATE, CD and COMP metric curves for main parameters in our approach.

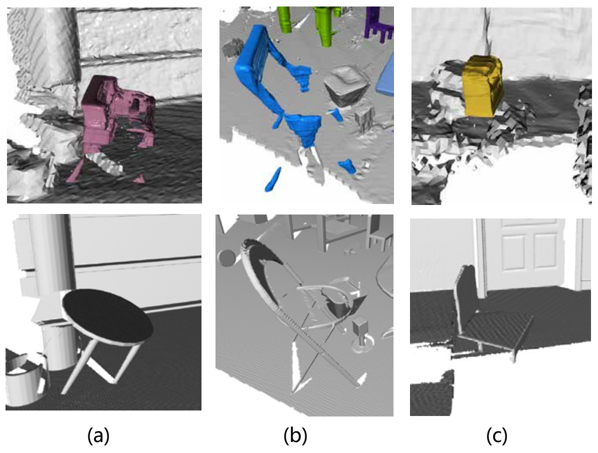

Figure 8. Failure cases of our system, such as objects that stand obliquely on the ground, thin parts of object, and those mostly occluded by other object.

and the last column demonstrates that the object is only partially reconstructed due to incomplete segmentation and is hard to rectify through optimization.

Parameters. We make a parameter study to evaluate some key parameters used in our system including 'good' object threshold τ_{ok}, 'failed' object threshold τ_{fail}, and the silhouette weight w_1. In the parameter evaluation, we first sample τ_{ok} in $\{0.001, 0.002, 0.005, 0.008, 0.01\}$, τ_{fail} in $\{0.01, 0.02, 0.04, 0.08, 0.1\}$ and w_1 in $\{0.1, 0.2, 0.5, 0.8, 1.0\}$, and then study the impact of each parameter on the accuracy of the final result one by one. Fig. 7 shows the ATE, CD and COMP accuracy curves on average for the three parameters tested on SceneNet RGB-D dataset we created separately. For parameter τ_{ok}, the ATE error increases consistently along with the τ_{ok} increasing. For parameter τ_{fail}, the CD metric increase and COMP metric decrease consistently along with the τ_{fail} increasing. For silhouette weight parameter w_1, when weight for silhouette term increases, lower chamfer distance will be achieved, but completeness metric is not well related to the parameter.

4.7. Limitations

There are some limitations of our approach: (1) we make the assumption that all objects are placed vertically on the ground, which will benefit for making good initialization of pose estimation. This may lead to poor performance when estimating and reconstructing those standing obliquely on the ground due to bad pose initialization, as Fig. 8(a) shows. However, since the assumption of vertically placed objects meets most object cases in indoor scenes, we think such assumption is still a reasonable point and leave its drawbacks as a limitation for future work. One possible solution would be using a deep model to predict initial pose guess without any constraint. (2) Thin structure objects. The thin structure objects would be challenging to detect completely from 2D CNN, which would not be correctly generated by our approach, e.g. the thin legs and seating part of the chair shown in Fig. 8(b). (3) The wrong shape generation caused by occlusion or truncation. If an object is partially observed with occlusion by other objects, our approach could not give accurate initial shape estimation, and thus incapable of optimizing for the final accurate shape generation(Fig. 8(c)).

5. Conclusion

In this paper, we present a novel object-level RGB-D SLAM system by using deep neural implicit representation for objects. Our method can effectively learn object shape prior and encode object as a latent vector for accurate and complete object reconstruction while saving large memory costs. Benefiting from the learned neural object priors, our proposed accurate camera tracking and joint optimization for object shape, object pose and camera pose for the final high-quality object surface reconstruction. We have shown our approach achieves better performance for accurate camera/object pose estimation and high quality of object shape reconstruction, evaluated both on synthetic and real-scan scenes. In the future, we hope to further improve our system to solve the drawbacks aforementioned for a better object-level SLAM with more robust camera tracking and better object surface reconstruction quality, or even in dynamic scenes.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant No. 61902210).

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics (TRO)*, 32(6):1309–1332, 2016. 2

[2] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal. SDF tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In *IROS*, pages 3671–3676, 2013. 5

[3] Y.-P. Cao, L. Kobbelt, and S.-M. Hu. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Trans. Graph.*, 37(5):171:1–171:16, 2018. 1, 2

[4] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. A. Newcombe. Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In *ECCV*, volume 12374 of *Lecture Notes in Computer Science*, pages 608–625, 2020. 1, 3

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, and H. Su. Shapenet: An information-rich 3d model repository. *Computer Science*, 2015. 5, 7

[6] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.*, 32(4):113:1–113:16, 2013. 2

[7] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 7

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, pages 303–312, 1996. 1, 2

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE CVPR*, pages 2432–2443, 2017. 2, 7, 10

[10] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.*, 36(3):24:1–24:18, 2017. 1, 2, 7, 10

[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE T-PAMI*, 29(6):1052–1067, 2007. 2

[12] Z.-J. Du, S.-S. Huang, T.-J. Mu, Q. Zhao, R. Martin, and K. Xu. Accurate dynamic slam using crf-based long-term consistency. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2020. 2

[13] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE T-PAMI*, 40(3):611–625, 2018. 2

[14] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: large-scale direct monocular SLAM. In *ECCV*, pages 834–849, 2014. 2

[15] Y. Fang, S. Yang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu. Instances as queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6910–6919, October 2021. 10

[16] R. Gomez-Ojeda, F. A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. G. Jiménez. PL-SLAM: A stereo SLAM system through the combination of points and line segments. *IEEE Trans. Robotics*, 35(3):734–746, 2019. 2

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 10

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[19] J. Huang, S. Huang, H. Song, and S. Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *IEEE CVPR*, pages 8932–8941, 2021. 1, 3, 5

[20] S. Huang, Z. Ma, T. Mu, H. Fu, and S. Hu. Lidar-monocular visual odometry using point and line features. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 1091–1097. IEEE, 2020. 2

[21] S.-S. Huang, H. Chen, J. Huang, H. Fu, and S.-M. Hu. Real-time globally consistent 3d reconstruction with semantic priors. *IEEE transactions on visualization and computer graphics*, December 2021. 2

[22] C. M. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. A. Funkhouser. Local implicit grid representations for 3d scenes. In *IEEE CVPR*, pages 6000–6009, 2020. 1, 3

[23] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE TVCG*, 21(11):1241–1250, 2015. 2, 7

[24] Y. Li, S. Hao, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas. Joint embeddings of shapes and images via cnn image purification. *ACM Transactions on Graphics*, 34(6cd):234.1–234.12, 2015. 7

[25] H. Liu, G. Zhang, and H. Bao. Robust keyframe-based monocular SLAM for augmented reality. In W. Broll, H. Saito, and J. E. S. II, editors, *IEEE ISMAR*, pages 1–10, 2016. 2

[26] L. Liu, X. Xia, H. Sun, Q. Shen, J. Xu, B. Chen, H. Huang, and K. Xu. Object-aware guidance for autonomous scene reconstruction. *ACM Transactions on Graphics (TOG)*, 37(4):104:1–104:12, 2018. 1

[27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 4

[28] J. McCormac, R. Clark, M. Bloesch, A. J. Davison, and S. Leutenegger. Fusion++: Volumetric object-level SLAM. In *3DV*, pages 32–41, 2018. 1, 2, 3

[29] J. Mccormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 7

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In A. Vedaldi,

H. Bischof, T. Brox, and J. Frahm, editors, *ECCV*, volume 12346, pages 405–421. Springer, 2020. 3

[31] N. Müller, Y. Wong, N. J. Mitra, A. Dai, and M. Nießner. Seeing behind objects for 3d multi-object tracking in RGB-D sequences. In *IEEE CVPR*, 2021. 1

[32] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957. 4

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015. 6

[34] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 2, 7

[35] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011. 1

[36] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, 2013. 1, 2

[37] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE CVPR*, pages 165–174, 2019. 1, 4, 5, 7

[38] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE CVPR*, pages 165–174, 2019. 3

[39] S. Peng, M. Niemeyer, L. M. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *ECCV*, volume 12348, pages 523–540. Springer, 2020. 3

[40] V. A. Prisacariu, A. V. Segal, and I. Reid. Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *Asian conference on computer vision*, pages 593–606. Springer, 2012. 5

[41] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *ISMAR*, pages 10–20, 2018. 1, 3, 7

[42] M. Runz, K. Li, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, et al. Frodo: From detections to 3d objects. In *IEEE CVPR*, pages 14720–14729, 2020. 1, 3, 5, 7, 9

[43] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: simultaneous localisation and mapping at the level of objects. In *IEEE CVPR*, pages 1352–1359, 2013. 1, 3

[44] M. Strecke and J. Stuckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2019. 3

[45] E. Sucar, S. Liu, J. Ortiz, and A. Davison. iMAP: Implicit mapping and positioning in real-time. In *International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 3, 4

[46] E. Sucar, K. Wada, and A. Davison. Nodeslam: Neural object descriptors for multi-view shape reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 949–958. IEEE, 2020. 1, 2, 4, 5, 7, 9, 10

[47] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *IEEE CVPR*, pages 15598–15607, 2021. 3

[48] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense SLAM and light source estimation. *I. J. Robotics Res.*, 35(14):1697–1716, 2016. 2

[49] Y. Wong, C. Li, M. Nießner, and N. J. Mitra. Rigidfusion: RGB-D scene reconstruction with rigidly-moving objects. *Computer Graphics Forum (CGF)*, 40(2):511–522, 2021. 1, 3

[50] Y. Xiao, Y. Lai, F. Zhang, C. Li, and L. Gao. A survey on deep geometry learning: From a representation perspective. *Comput. Vis. Media*, 6(2):113–133, 2020. 3

[51] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. J. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic SLAM. In *IEEE ICRA*, pages 5231–5237, 2019. 1, 2, 3, 7

[52] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Transactions on Graphics (TOG)*, 34(6):177:1–177:14, 2015. 1