2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Compositional and Scalable Object SLAM

Akash Sharma, Wei Dong, and Michael Kaess

***Abstract*— We present a fast, scalable, and accurate Simultaneous Localization and Mapping (SLAM) system that represents indoor scenes as a graph of objects. Leveraging the observation that artificial environments are structured and occupied by recognizable objects, we show that a compositional and scalable object mapping formulation is amenable to a robust SLAM solution for drift-free large-scale indoor reconstruction. To achieve this, we propose a novel semantically assisted data association strategy that results in unambiguous persistent object landmarks and a 2.5D compositional rendering method that enables reliable frame-to-model RGB-D tracking. Consequently, we deliver an optimized online implementation that can run at near frame rate with a single graphics card, and provide a comprehensive evaluation against state-of-the-art baselines. An open-source implementation will be provided at `https://github.com/rpl-cmu/object-slam`.**

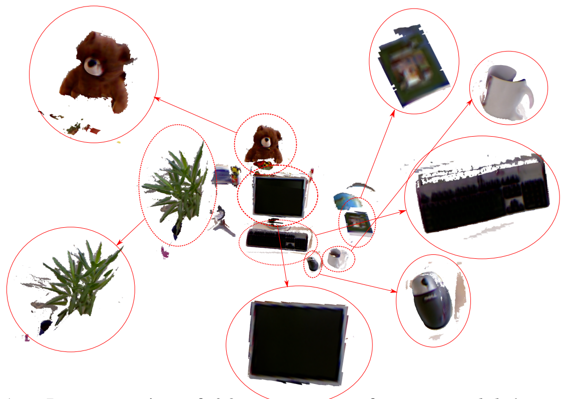
Fig. 1. Reconstruction of *fr2_xyz* sequence from *tum rgbd* dataset. Our pipeline can reconstruct both camera trajectory and object models in the scene.

I. Introduction

Autonomous robots that work in the real world require advanced interpretation of the world, from semantic 3D reconstruction, path planning, to active interaction with the environment. These workloads require not only geometric perception including robot localization and dense scene reconstruction, but also semantic and compositional understanding of scenes.

In recent years, geometry-based SLAM has achieved high levels of performance in *experimental setups* for localization tasks. Many variants of SLAM algorithms, from ORB-SLAM [1] to Direct Sparse Odometry (DSO) [2], can now run in real-time with high trajectory accuracy. However, they are in general limited by the *static-world* assumption and low-level scene representation (sparse 3D feature points), and thus cannot distill high-level information (semantic understanding) in scenes and adjust to structured environmental changes.

On the other hand, with progress in deep learning, near frame rate semantic perception is achievable powered by efficient Deep Neural Networks (DNNs). Researchers have started to switch to semantic SLAM taking advantage of off-the-shelf solutions; pioneering research includes SLAM++ [3], Fusion++ [4], and MaskFusion [5]. These initial attempts take into consideration semantic segmentation, but typically simply attach DNN frontends to existing SLAM frameworks in an ad hoc fashion. Implementation-wise, they require high-end machines to achieve near real-time performance, or are not available to the community.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {akashsha, weidong, kaess}@andrew.cmu.edu

This work was partially supported by ARL award W911NF-17-2-0181.

To address these problems, we propose a novel modular solution that concentrates on recognizable persistent object landmarks. In theory, we derived a compositional and scalable object map for robust tracking. In implementation, we fully exploit the power of the modern GPU-based reconstruction pipeline [6] and object detection frameworks [7], to design an efficient architecture for data exchange without sacrificing the ease of system configuration and build.

Our main contributions in this paper are:

1) A compositional volumetric rendering method that selects objects of interest and reduces memory footprint;
2) A hybrid object association method that combines geometric and semantic cues to enable drift-free tracking without an explicit relocalization module;
3) A scalable, modular, and easy-to-use open source system that runs nearly realtime.

II. Related Work

In this section we review relevant literature in two aspects: classical geometry-based SLAM, and the application of deep semantic object detection in SLAM.

A. Geometry-based SLAM

Problem formulation and pose optimization: Modern geometry-based SLAM systems can be generally classified into *feature-based* and *direct* methods. Feature-based SLAM systems [1], [8] usually maintain a collection of sparse 3D *point landmarks* corresponding to hand-crafted feature *keypoints* detected in 2D images. In order to correct accumulated *pose* error, *i.e.,* drift, these methods resort to bundle adjustment [9] that jointly minimizes reprojection error between *landmarks* and 2D *keypoints* via *pose* optimization. While

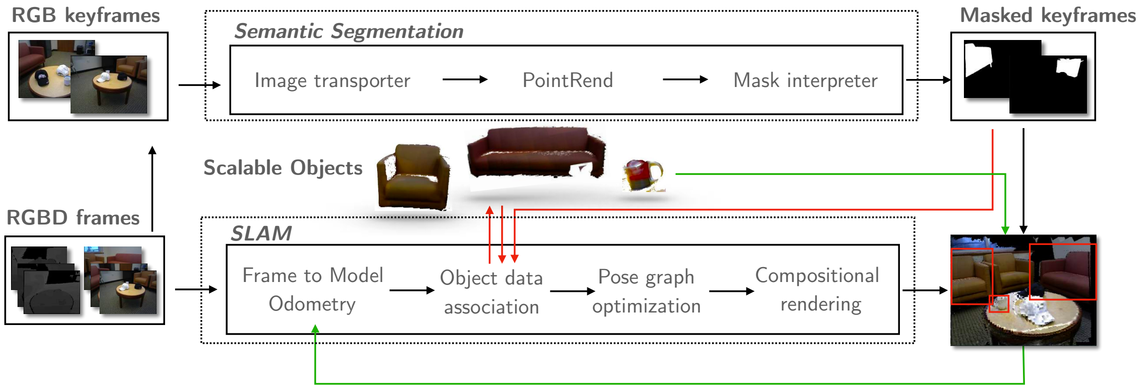

Fig. 2. System overview: Top shows the deep object segmentation pipeline that runs asynchronously, Masked keyframes from the segmentation pipeline are used in Data association and map update (shown with red lines). Bottom shows the major stages of the reconstruction system, specifically object models are used in tracking via compositional raycasting (shown with green lines).

being accurate in estimating trajectory, these approaches only come with sparse 3D maps that are less interpretable for visualization and recognition. Direct SLAM [10], [2] on the other hand, relies on pixel-wise projective data association between frames for odometry. Given relative poses between certain keyframes, pose graphs [11] are formulated and optimized to obtain globally consistent poses without landmark constraints.

Our approach can be regarded as a bridge between the two approaches. We replace point landmarks with objects in feature-based SLAM. As a result, since pose constraints attached to objects can naturally replace reprojection error, we may directly convert such a landmark-pose constraint optimization to pose graph optimization (PGO).

Map representation: For *dense* scene reconstruction, the volumetric Truncated Signed Distance Function (TSDF) [12] representation has been adopted and improved in several *direct SLAM* frameworks. KinectFusion [13] introduced a plain 512^3 grid for small scenes and objects. VoxelHashing [14] designed spatial hashing to scale this data structure to larger scenes. Similar implementations are available in CPU/GPU in the modern Open3D framework [15], [6], and we adopt this representation due to its ease of use.

B. Object instance segmentation and object-based SLAM

Object instance segmentation: In recent years, *Region proposal* based Convolutional Neural Networks (R-CNN) [16] have established themselves as de-facto standards for object *instance segmentation* from images. Amongst the literature, *Mask-RCNN* [17] and *PointRend* [7] are the best off-the-shelf solutions. In this work, we use *PointRend* [7], which shows significant improvement over [17] by reformulating the mask generation as a rendering problem. In essence, this formulation is consistent with our tracking via compositional rendering module.

Object-based SLAM: Applying aforementioned DNNs on 2D images, several works for RGB-D and monocular SLAM have attempted to incorporate object instance detection. CubeSLAM [18] and QuadricSLAM [19] fit cuboids and quadrics, respectively, to detected objects to generate parameterized object landmarks. While improving the localization accuracy compared to baselines, these methods fail to densely map objects. *MaskFusion* [5] adds labels to oriented point clouds and supports dense object visualization, but does not maintain persistent objects globally in a graph. *Fusion++* [4], on the other hand, supports persistent dense reconstruction from fixed size 64^3 voxel grids, yet is sensitive to voxel size tuning and may fail to adapt to objects at varying scales. Our system utilizes scalable voxel grids that do not require much tuning to adjust to object scales. With a seamless CPU to GPU memory transfer implementation, larger environments can also be handled on-the-go.

III. METHOD

Our pipeline can be divided into typical SLAM components and a deep perception module, connected by an object-based semantic map. Figure 2 provides an overview.

It consists of 5 modules each running in a separate thread: semantic segmentation, frame-to-model odometry, object data association and map update, PGO, and compositional rendering. Incoming *RGB-D* frames are initially processed through *semantic segmentation* (§III-C) to obtain instance masks, labels, and semantic descriptors, from DNNs for keyframes. Then, odometry between the incoming live frame and the *compositional render* from the map (§III-E) is estimated via *frame-to-model odometry* (§III-B) to obtain relative poses. Maintained objects visible in the frame are rendered given the estimated camera pose, and objects are associated with 2D instance detections to either integrate or initialize new objects in the global map (§III-C). Separately, a global factor-graph is updated to optimize the camera trajectory and object poses (§III-D). The optimized object poses are rendered to generate a compositional model of the scene for subsequent tracking (§III-E).

Before we discuss these modules in detail from §III-B to §III-E, we introduce core concepts and notations in §III-A.

A. Core concepts and notations

A background volume V_B is a spatially-hashed voxel grid on GPU, where small 16^3 subvolumes are allocated around observed 3D points. It is created and updated as a *temporary* instance for stable tracking. An object volume V_{O_i} is akin to the background volume, but persistently maintains the object label, ID, and corresponding object descriptors.

A 3D volume V's properties, including surface vertex positions, normals, and colors, can be mapped to 2D images given a camera pose $\mathbf{T} \in SE(3)$ and camera intrinsics defined as K with ray-casting. We denote such *rendered images* by $\langle \mathcal{N}, \mathcal{V}, \mathcal{C} \rangle$ for normal, vertex, and color maps respectively. They can be associated with *input RGB-D images* $\langle \mathcal{I}, \mathcal{D} \rangle$ that consist of color ($\mathcal{I}$) and depth ($\mathcal{D}$) images via projective closest points.

We use subscripts and superscripts to indicate multiple coordinate frames used in our pipeline, including C_i for ith camera, O_j for jth object, and W for background or world coordinate frame. For instance, $\langle \mathcal{N}_{C_s}, \mathcal{V}_{C_s}, \mathcal{C}_{C_s} \rangle$ represents 2D maps rendered from the volumes in the C_s camera coordinate frame. $\mathbf{T}_{O_j}^{W} \in SE(3)$ encodes a rigid transformation from object j to world. Finally, we denote the respective measurements between nodes with variable $\mathbf{Z}$.

B. Hybrid frame-to-model odometry

In *RGB-D* camera tracking, we seek to estimate the relative camera pose $\mathbf{T}_{C_s}^{C_t}$ given an incoming *RGB-D* target frame $\langle \mathcal{I}_{C_t}, \mathcal{D}_{C_t} \rangle$ and a source model $\langle \mathcal{N}_{C_s}, \mathcal{V}_{C_s}, \mathcal{C}_{C_s} \rangle$ of the scene rendered by placing a virtual camera at the previous camera frame C_s.

We accomplish this by minimizing the joint weighted dense geometric error residual r_D and the photometric error residual r_I. The general energy function is formulated as in [20] by accumulating residual at every point $p \in \mathbb{R}^2$ with a valid data association:

$$E(\mathbf{T}_{C_s}^{C_t}) = \sum_{p} (1-\sigma) r_I^2(\mathbf{T}_{C_s}^{C_t}, p) + \sigma r_D^2(\mathbf{T}_{C_s}^{C_t}, p), \quad (1)$$

Here, we adapt the geometric ICP residual as the point-to-plane distance between the incoming depth map $\mathcal{D}$ and the rendered vertex and normal map $(\mathcal{V}_{C_s}, \mathcal{N}_{C_s})$ as follows, using the formulation in [13]:

$$r_D(T_{C_s}^{C_t}, p) = \Big((T_{C_s}^{C_t} \mathcal{V}_{C_s}(\hat{p}) - \mathcal{V}_{C_t}(p) \Big) \cdot \mathcal{N}_{C_t}(p), \quad (2)$$

where $\mathcal{V}_{C_t}$ is the vertex map from unprojecting the input depth image $\mathcal{D}_{C_t}$. Additionally, we use a photometric error residual to improve tracking robustness, which is defined as:

$$r_I(T_{C_s}^{C_t}, p) = \mathcal{C}_{C_s}(\hat{p}) - \mathcal{I}_{C_t}(p). \quad (3)$$

In equations (2) and (3), $\hat{p}$ is the correspondence of p in the source frame, and is computed via *warping*:

$$\hat{p} = K {\mathbf{T}_{C_s}^{C_t}}^{-1} \mathcal{D}_{C_t}(p) K^{-1} [p^\top, 1]^\top. \quad (4)$$

It must be noted that the ps are a subset of pixels with valid object-level data associations detailed in §III-E.

The energy function in equation 1 is minimized using the Gauss-Newton algorithm. We implement the minimization in a coarse to fine scheme using an image pyramid, on the GPU in parallel since each pixel acts independently in the energy function using *reduction* with appropriate thread conflict handling as described in [6].

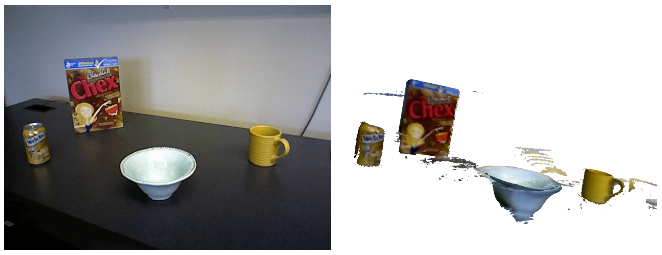
Fig. 3. Qualitative foreground object reconstruction results on *RGB-D Scene 13* sequence.

C. Object instance segmentation and association

2D instance segmentation: Object detection and instance masks are generated every n^{th} frame (we choose $n = 10$) in a separate thread from the *PointRend* backend. *PointRend* uses a Resnet-50-FPN backbone network to generate a convolutional feature map. In particular, after an empirical evaluation, we found that *PointRend* provided better masks over *Mask-RCNN*.

The semantic segmentation module maps incoming *RGB* frame $\mathcal{I}$ into a set of object labels $[l_1, \dots l_k]$, a set of binary object masks M_n^i defined over $l \in \mathcal{L} \triangleq \{0, \dots, L_{max} - 1\}$ object classes ($L_{max} = 80$ in the MS-COCO dataset), bounding boxes $b \in \mathbb{N}^4$, and a probability distribution $p(l_i \mid \mathcal{I})$. We also extract the object feature map for the accepted object proposals, from the penultimate fully connected layer of the R-CNN from the object classifier head. We observe that these feature maps provide us with robust data association in ambiguous situations. To obtain instance segmentation for frames not sent to the DNN, we warp the binary mask images from the most recent frame with a segmentation and fill the holes in the warped masks using the *flood fill* algorithm.

Once the current camera pose and the semantic segmentation information are available, instance detections are associated with existing objects. Unmatched instance detections are used to initialize new object volumes.

3D instance generation: When an unmatched object is to be instantiated, the masked depth frame at C_i is unprojected and transformed into the world frame to obtain the object point cloud:

$$X_W = \mathbf{T}_{C_i}^{W} K^{-1} D_{C_i}(p) [p^\top, 1]^\top. \quad (5)$$

To obtain relatively high fidelity reconstruction, we adaptively calculate a conservative voxel length of

$$l = \gamma \| \max(X_W) - \min(X_W) \|_\infty, \quad (6)$$

where $\min, \max$ operators are applied to all dimensions of $X \in \mathbb{R}^3$ simultaneously. We empirically use $\gamma = 1/64\sqrt{2}$, but due to the scalability of the volume our model is less sensitive to γ. Finally, the object pose is simply chained by

$$\mathbf{T}_{O}^{W} = \mathbf{T}_{C_i}^{W} \Big(\mathbf{T}_{C_i}^{O} \Big)^{-1}, \quad (7)$$

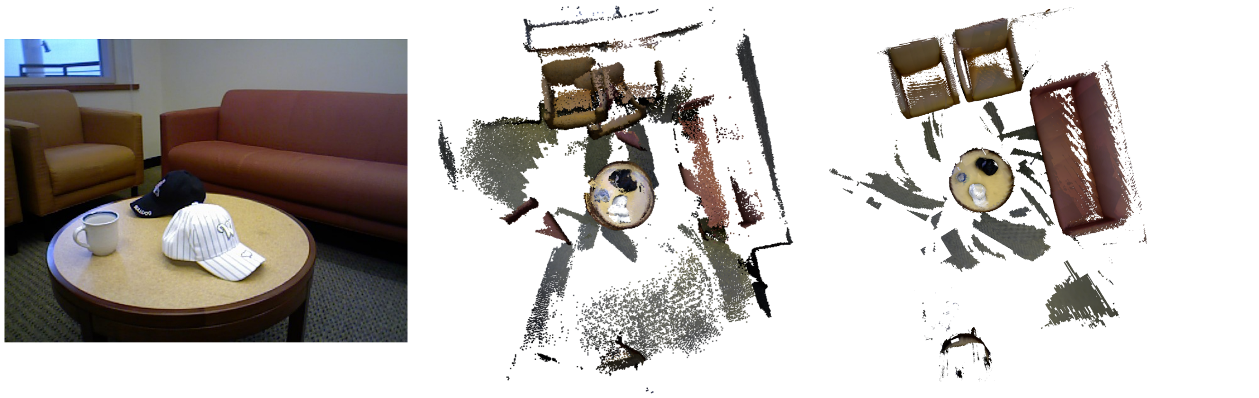

Fig. 4. Reconstructed small indoor scene *RGB-D Scene 12*. We first show an example input *RGB* frame followed by a top-down view of the reconstruction from *MaskFusion*. This is followed by result from our pipeline. Note that in our reconstruction background walls and floor are filtered out.

where $\mathbf{T}^{O}_{C_i} = [I \mid t^{O}_{C_i}]$ with $t^{O}_{C_i} = \min(X_W) - t^{W}_{C_i}$. Each new object is also initialized with the object feature map from its corresponding instance mask.

2D–3D semantic data association: To associate existing object volumes to 2D instances, visible objects are rendered (in §III-E) in the current frame. The rendered color map $\mathcal{C}$ is thresholded to obtain a virtual binary mask. An intersection over union (IoU) between the virtual binary mask $\hat{\mathcal{M}}$ and the instance masks $\mathcal{M}_i$ in the current frame is used as a scoring metric as defined in [4].

As opposed to computing the $\operatorname{argmax}_i \mathrm{IoU}(\mathcal{M}_i, \hat{\mathcal{M}})$, we associate objects as given below:

$$i = \underset{i \in \mathcal{S}}{\operatorname{argmin}}(\|f_i - \hat{f}\|_1), \tag{8}$$

where $\hat{f}$ and f_i denote feature map of the object render (identical to the object in question), and the instance masks respectively and $\mathcal{S} \triangleq \{i : \mathrm{IoU}(\mathcal{M}_i, \hat{M}) > 0.2\}$. Associating object renders to instance masks in this manner prevents incorrectly fusing object instances between nearby similar objects, in cases where there is large accumulated drift.

For subsequent fusion of a 2D instance detection to its associated 3D object, the instance mask—containing the object foreground—and the bounding box mask—containing both the foreground and background are used. Similar to [4] we integrate the object in both the foreground and background through a weighted average of TSDF, color, and additionally maintain binomial foreground-background count variables for each voxel. This smoothes out artifacts from integration of 2D instances with spurious masks.

Finally, we update the object feature map by a gated weight average:

$$f_t = \frac{w_{t-1} \cdot f_{t-1} + \mathcal{H}(f_{t-1}, f_{in}) \cdot f_{in}}{w_{t-1} + \mathcal{H}(f_{t-1}, f_{in})}, \tag{9}$$

$$\mathcal{H}(f_{t-1}, f_{in}) = \frac{\mathrm{sgn}(\lambda - ||f_{t-1} - f_{in}||_1) + 1}{2}, \tag{10}$$

where $\mathcal{H}$ is the Heaviside step function that hard-filters outlier input feature map f_{in} compared to the maintained object feature map f_{t-1} with weight w_{t-1} controlled by the threshold λ.

D. Factor graph optimization

As we have mentioned before, a background volume is maintained for stable tracking, and to handle *objectless* frames. The background volume, additionally maintains the ratio (r) of visible volume units in the current camera frustum to the total number allocated volume units in the volume. A low ratio implies that the camera may have moved away from a particular part of the scene. Pose graph optimization is conditionally triggered when the background volume is reset owing to low ratio of visible units ($r < 0.2$) and when there are new objects added into the graph.

Our object factor graph formulation is similar to [3], [4]. The variable nodes $\mathcal{X} = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ are partitioned into camera pose variables $\mathbf{T}^{W}_{C_i} \in SE(3)$ and object pose variables $\mathbf{T}^{W}_{O_j} \in SE(3)$. The first camera pose is initialized as the world frame W.

Assuming a Gaussian noise model, the *MAP* inference problem with the above variable nodes reduces to solving the following non-linear least squares optimization:

$$\begin{aligned}\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \Big(&\sum_{k \in |C|} \|\mathbf{Z}^{C_k}_{C_{k-1}} \ominus \mathbf{T}^{C_k}_{C_{k-1}}\|^2_{\Sigma_{k,k-1}} \\ &+ \sum_{j \in |\mathcal{O}|, k \in |\mathcal{C}|} \|\mathbf{Z}^{O_j}_{C_k} \ominus \mathbf{T}^{O_j}_{C_k}\|^2_{\Sigma_{o_j,k}} \Big), \end{aligned} \tag{11}$$

where the operator $\mathcal{Y} \ominus \mathcal{X} = \mathrm{Log}(\mathcal{X}^{-1}\mathcal{Y})$ expresses the relative error in the local tangent vector space [21]. $\Sigma_{k,k-1}$ denotes the covariance between relative camera pose measurements, $\Sigma_{o_j,k}$ is the covariance in the camera to object measurement. They can be approximated by information matrices computed from *odometry*, however, empirically we found that a constant information matrix can achieve reasonable results. We obtain the relative camera measurements $\mathbf{Z}^{C_k}_{C_{k-1}}$ from *frame to model odometry* (§III-B), and obtain frame to object measurements $\mathbf{Z}^{O_j}_{C_k}$ by performing an additional Gauss Newton iteration with only the object pixels. Finally, the expected relative camera pose $\mathbf{T}^{C_k}_{C_{k-1}}$ and expected camera object pose $\mathbf{T}^{O_j}_{C_k}$ used in the factors are

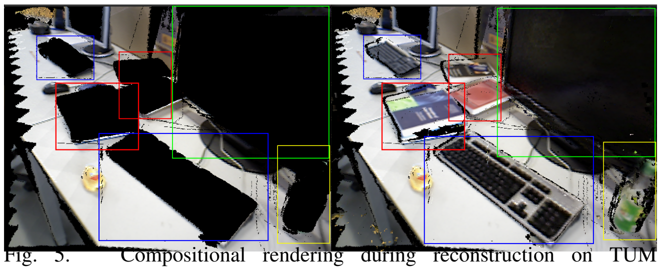
Fig. 5. Compositional rendering during reconstruction on TUM *fr3_long_office_household* dataset. Left shows the background render, and right shows the composed render. Compositionally rendered objects are shown in bounding boxes.

calculated as:

$$\mathbf{T}_{C_{k-1}}^{C_k} = \left(\mathbf{T}_{C_k}^{W}\right)^{-1} \mathbf{T}_{C_{k-1}}^{W}, \tag{12}$$

$$\mathbf{T}_{C_k}^{O_j} = \left(\mathbf{T}_{O_j}^{W}\right)^{-1} \mathbf{T}_{C_k}^{W}. \tag{13}$$

We solve the optimization in GTSAM [22], using Levenberg Marquardt. Since the entire object volume is transformed as a rigid body, the object volumes remain unchanged in memory after optimization. We note that this circumvents the time-consuming re-integration that usually takes place in volumetric methods after PGO [23].

E. Compositional rendering

Compositional rendering is a serialized operation that generates normal, vertex, and color maps by ray-casting 3D objects in the viewing frustum into 2D object instances, and is illustrated in Figure 5.

$\langle \mathcal{N}_{C_i}, \mathcal{V}_{C_i}, \mathcal{C}_{C_i} \rangle$ is in fact an aggregation of separate renderings from object volumes $\langle \mathcal{N}_{C_j}^{V_{O_j}}, \mathcal{V}_{C_i}^{V_{O_j}}, \mathcal{C}_{C_i}^{V_{O_j}} \rangle$ and the background volume $\langle \mathcal{N}_{C_i}^{V_B}, \mathcal{V}_{C_i}^{V_B}, \mathcal{C}_{C_i}^{V_B} \rangle$, depending on the masks. In particular, we render the background volume, based on a background mask that is constructed from the union of existing virtual object masks in the current frame, and associated instance masks.

Then, the composed per-pixel map model render can be obtained as follows:

$$\hat{k} = \operatorname*{argmin}_{k} \mathcal{V}_k(p)[z], \;\; k \in \{O_1, \cdots, O_n, B\} \tag{14}$$

$$\langle \mathcal{N}^*(p), \mathcal{V}^*(p), \mathcal{C}^*(p) \rangle = \langle \mathcal{N}_{\hat{k}}(p), \mathcal{V}_{\hat{k}}(p), \mathcal{C}_{\hat{k}}(p) \rangle, \tag{15}$$

where $\hat{k}$ is the volume index corresponding to the minimum distance to camera center for pixel p.

Object volumes not currently visible are downloaded from GPU into CPU memory. Note that downloading the object volume does not affect the optimization problem, since the object volumes are required only for integration and raycasting.

IV. System Architecture

To support relatively high frame rate operation in the presence of slow/non-realtime deep learning components our pipeline is highly parallelized. Our system adopts the *Actor* framework, where each component runs asynchronously, and communicates via thread-safe queues.

We implement the semantic segmentation pipeline as a separate python process which serializes the outputs using `protobuf` and communicates with the client thread via `zeromq` sockets in the Object SLAM pipeline. Since, instance segmentation is carried out only for keyframes, typically, the asynchronous python process exits early freeing GPU memory for larger scene reconstructions.

The GPU code is implemented in CUDA, and we leverage the Open3D framework [6] for a scalable TSDF implementation.

V. Experimental Results

In this section, we show that our system achieves comparable results to state-of-the-art online/offline reconstruction systems in terms of trajectory accuracy while being able to segment and reconstruct objects. We evaluate on the RGBD scenes V2 dataset [24] and TUM RGBD dataset [25], both of which are established RGBD benchmarks and compare against baselines. Our experiments were run on a Linux system with Intel i7-6700 CPU at 4.00GHz and 32GB of RAM and a NVIDIA GTX1080 with 8GB of GPU memory.

A. Qualitative results

We first demonstrate qualitative reconstruction results on the RGBD scenes V2 dataset. Fig. 3 shows the object mesh extracted from our scalable volumes with a foreground count threshold. We can see that small objects are clearly reconstructed with details, and the background is correctly filtered.

At a larger scale, Fig. 1 segments teddy bear and computers from the cluttered scene and ensures a low-drift of the trajectory. Fig. 4 compares reconstructions from *MaskFusion* [5] and our system for a given sequence. It can be seen that the object-level reconstruction, specifically for caps and sofas, is much cleaner by our system than by *MaskFusion*.

B. Quantitative results

Table I presents Absolute Trajectory Error (ATE) of four different methods compared with our system. Note in the table, the best results of *the object-based systems* are in *bold*, and provide results from geometric SLAM systems for reference.

In general, we achieve comparable results against the state-of-the-art surfel based online SLAM system *ElasticFusion* [23] and volumetric offline reconstruction system *Open3D* [15]. In the meantime, our method outperforms object-based SLAM systems *MaskFusion*[1] [5] and *Fusion++* [2][4] by a large margin. This improvement can be attributed to the use of semantic data association and scalable voxel grids.

For small scenes in the *RGB-D scenes V2 dataset*, we achieve consistently high accuracy with ATE below $5cm$ for all scenes. Figure 6 shows detailed trajectory visualizations.

[1]MaskFusion requires 2 high end graphics cards to run it in online mode. We ran it in offline mode, and stored all the detected object instances from the method instead of manually selecting objects of interest for a fair comparison with our method

[2]Fusion++ is not open sourced and we obtained the results from the paper.

TABLE I

TRAJECTORY ACCURACY COMPARISON ON REALWORLD DATASET (ABSOLUTE TRAJECTORY ERROR IN CENTIMETERS)

Dataset	ElasticFusion	Open3D	MaskFusion	Fusion++	Ours
Online	✓		✓	✓	✓
Object Models			✓	✓	✓
RGBD Scenes - Scene 03	1.42	19.37	26.67	-	**4.52**
RGBD Scenes - Scene 12	0.64	1.97	10.81	-	**2.36**
RGBD Scenes - Scene 14	1.09	1.33	8.26	-	**2.37**
freiburg1_xyz	6.33	6.64	8.68	-	**7.50**
freiburg1_desk	2.70	5.73	24.05	**4.9**	5.82
freiburg1_desk2	7.12	7.65	21.5	15.3	**9.57**
freiburg1_room	22.06	5.65	52.4	23.5	**21.7**
freiburg2_xyz	1.12	2.18	12.30	**2.0**	2.27
freiburg2_desk	7.61	4.72	163.6	11.4	**9.94**
freiburg3_long_office	2.23	3.54	140.8	10.8	**9.68**

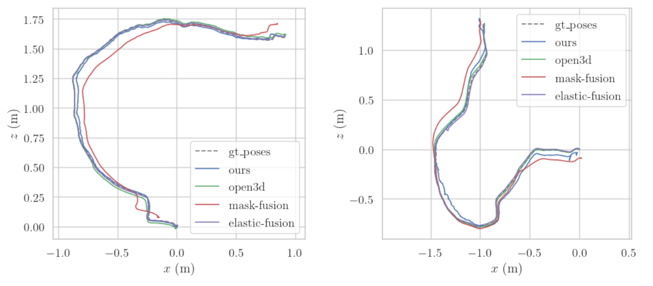

Fig. 6. Comparison of trajectories between our pipeline and baselines with ground truth. (a) shows *rgbd-scenes-v12* and (b) shows *rgbd-scenes-v14* sequences.

For larger scenes, although noisy semantic segmentations affect masks and introduce noise for frame-to-model odometry, compositional rendering still ensures reliable tracking. Trajectory comparisons are provided in Figure 7.

C. Runtime analysis

For runtime evaluation of our system we limit the number of initialized objects in the scene to 10 to ensure (close to) online performance on the aforementioned scenes. Processing each frame in the absence of any objects i.e., only background tracking takes about 200ms per frame. The largest computational bottleneck and time consuming operation is the rendering step, and while there are multiple rendering operations required (for instance, during object association), we render the objects and background only once per frame, and reuse the renders. We observe that each object takes on average about 45ms to render. In the presence of about 5-8 objects in the scene, the time taken per frame increases to about 450ms (200ms for background + 250ms for object renders) per frame. In comparison, we observed in our tests that *MaskFusion* runs at lower than 1FPS with one graphics card and suffers from random crashes in online mode. *Fusion++* reports its results with pre-computed segmentation masks. Our method runs seamlessly on a single GPU. A detailed runtime analysis is given in Table II.

TABLE II

RUNTIME BREAKDOWN COMPONENT-WISE FOR OUR PIPELINE

Component	Tracking	Segmentation	Association	Rendering
Time (ms)	13	250	15	45 per object

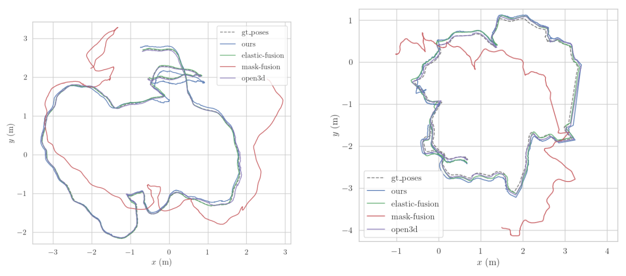

Fig. 7. Trajectory comparisons on the TUM-RGBD dataset (a) *fr3_long_office_household* and (b) *fr2_desk* sequences showing that even in the absence of explicit loop closures, our system maintains comparable accuracy.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented Compositional and Scalable Object SLAM, which bridges geometry-based techniques effectively with deep object detection. The system maintains persistent independent 3D models of the objects visible in the scene, which provides for relatively accurate trajectories as well as object reconstructions.

While our system makes significant progress towards semantic SLAM, it is not perfect owing to the following shortcomings. 1) TSDF inpainting is not considered, causing partially reconstructed objects; the system slows down subsequently with increasing map size, albeit at a slow rate. 2) Object labels are limited to 80 classes from the MS-COCO dataset; 3) Finally in the presence of instance switches of detected objects across time and missed detections of small objects, tracking accuracy is affected.

Solving these shortcomings provide avenues for interesting future work. In particular, we plan to introduce object model based reconstructions and neural rendering for better object reconstructions.

References

[1] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, arXiv: 1610.06475. [Online]. Available: http://arxiv.org/abs/1610.06475

[2] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[3] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*. Portland, OR, USA: IEEE, Jun. 2013, pp. 1352–1359. [Online]. Available: http://ieeexplore.ieee.org/document/6619022/

[4] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level SLAM," in *Proc. of International Conference on 3D Vision*. IEEE, 2018, pp. 32–41.

[5] M. Rünz, M. Buffier, and L. Agapito, "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects," Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.09194v2

[6] W. Dong, J. Park, Y. Yang, and M. Kaess, "GPU accelerated robust scene reconstruction," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7863–7870.

[7] A. Kirillov, Y. Wu, K. He, and R. Girshick, "PointRend: Image Segmentation as Rendering," *arXiv:1912.08193 [cs]*, Feb. 2020, arXiv: 1912.08193. [Online]. Available: http://arxiv.org/abs/1912.08193

[8] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, 2007, pp. 225–234.

[9] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[10] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[11] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017. [Online]. Available: http://www.nowpublishers.com/article/Details/ROB-043

[12] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer Graphics and Interactive techniques*, 1996, pp. 303–312.

[13] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. of IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.

[14] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–11, Nov. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508363.2508374

[15] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847 [cs]*, Jan. 2018, arXiv: 1801.09847. [Online]. Available: http://arxiv.org/abs/1801.09847

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[18] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[19] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 1–8, 2019.

[20] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored Point Cloud Registration Revisited," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017, pp. 143–152. [Online]. Available: http://ieeexplore.ieee.org/document/8237287/

[21] J. Sola, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[22] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," in *Technical Report*, 2012.

[23] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[24] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3050–3057, iSSN: 1050-4729.

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.