Fusion++: Volumetric Object-Level SLAM

John McCormac* Ronald Clark* Michael Bloesch Andrew J. Davison Stefan Leutenegger
Dyson Robotics Laboratory
Department of Computing, Imperial College London
{brendan.mccormac13,ronald.clark,m.bloesch,a.davison,s.leutenegger}@imperial.ac.uk

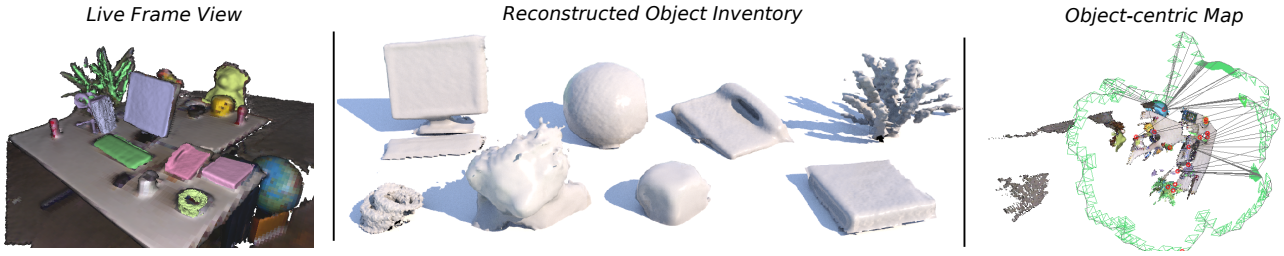

Figure 1: Fusion++ with pose graph and discovered inventory on the public fr2_desk sequence [33].

Abstract

We propose an online object-level SLAM system which builds a persistent and accurate 3D graph map of arbitrary reconstructed objects. As an RGB-D camera browses a cluttered indoor scene, Mask-RCNN instance segmentations are used to initialise compact per-object Truncated Signed Distance Function (TSDF) reconstructions with object size-dependent resolutions and a novel 3D foreground mask. Reconstructed objects are stored in an optimisable 6DoF pose graph which is our only persistent map representation. Objects are incrementally refined via depth fusion, and are used for tracking, relocalisation and loop closure detection. Loop closures cause adjustments in the relative pose estimates of object instances, but no intra-object warping. Each object also carries semantic information which is refined over time and an existence probability to account for spurious instance predictions.

We demonstrate our approach on a hand-held RGB-D sequence from a cluttered office scene with a large number and variety of object instances, highlighting how the system closes loops and makes good use of existing objects on repeated loops. We quantitatively evaluate the trajectory error of our system against a baseline approach on the RGB-D SLAM benchmark, and qualitatively compare reconstruction quality of discovered objects on the YCB video dataset. Performance evaluation shows our approach is highly memory efficient and runs online at 4-8Hz (excluding relocalisation) despite not being optimised at the software level.

*These two authors contributed equally.

1. Introduction

Indoor scene understanding and 3D mapping is a foundational technology that can enable autonomous real-world robotic task completion and also provide a common interface for more intelligent and intuitive human-map and human-robot interactions. To enable this requires a careful choice of map representation. One particularly useful representation is to build an object-oriented map. We argue this is a natural and efficient way to represent the things that are most important for robotic scene understanding, planning and interaction; and it is also highly suitable as the basis for human-robot communication.

In an object level map, the geometric elements which make up an object are grouped together as instances and can be labelled and reasoned about as units, in contrast to approaches which independently label dense geometry such as surfels or points. This approach also naturally paves the way towards interaction and dynamic object reasoning, although our system currently assumes a static environment and does not yet aim to track individual dynamic objects.

In this work we demonstrate an object-oriented online SLAM system with a focus on indoor scene understanding using RGB-D data. We aim to produce semantically labelled TSDF reconstructions of object instances without strong *a priori* knowledge of the object types present in a scene. We use Mask R-CNN [13, 40] to provide 2D instance mask predictions and fuse these masks online into the TSDF reconstruction (see Figure 1) along with a 3D 'voxel mask' to fuse the instance foreground (see Figure 3).

Unlike many dense reconstruction systems [24, 39, 43,

38, 3, 8] we make no attempt to keep a dense representation of the entire scene. Our persistent map consists of only reconstructed object instances. This allows the use of rigid TSDF volumes for high-quality reconstructions to be combined with the flexibility of a pose-graph system without the complication of performing intra-TSDF deformations. Each object is contained within a separate volume, allowing each one to have a different, suitable, resolution with larger objects integrated into lower fidelity TSDF volumes than their smaller counterparts. It also enables tracking large scenes with relatively small memory usage and high-fidelity reconstructions by excluding large volumes of free-space. A throw-away local TSDF of unidentified structure is used to assist tracking and model occlusions.

We capture a repeated loop of an indoor office scene to evaluate the system under conditions of occasional poorly constrained ICP tracking. The scene also contains a large number and variety of objects which not only exhibit the generality of the approach but is useful for evaluating the memory and run-time scaling of the method with many objects. While not optimised for real-time operation, we achieve $\sim$4-8Hz operating performance (excluding relocalisation/graph optimisation) on our office sequence and are confident that with sufficient optimisation true real-time operation is possible. We also quantitatively evaluate the trajectory error improvement of our system over a baseline approach on the RGB-D SLAM Benchmark [33].

In this work we make the following contributions:

- A generic object-oriented SLAM system which performs mapping as variable resolution 3D instance reconstruction.
- Per-frame instance detections are robustly fused using voxel foreground masks and missing detections are accounted for with an "existence" probability.
- We show high quality object reconstruction within globally consistent loop-closed object SLAM maps.

2. Related work

For reconstruction, we follow the TSDF formulation of Curless and Levoy [6] and the KinectFusion approach of Newcombe *et al.* [23] for local tracking. Our approach to object-level reconstruction is related to the work of Zhou and Koltun [42], where "points of interest" were detected and the aim was to reconstruct the scene so as to preserve detail in these areas while distributing drift and registration errors throughout the rest of the environment. In our work we analogously aim to optimise the quality of object reconstructions and allow residual error to be absorbed in the edges of the pose graph.

SLAM++ by Salas-Moreno *et al.* [30] was an early RGB-D object-oriented mapping system. They used point pair features for object detection and a pose graph for global optimisation. The drawback was the requirement that the full set of object instances, with their very detailed geometric shapes, had to be known beforehand and preprocessed in an offline stage before running. Stückler and Behnke [31] also previously tracked object models learned beforehand by registering them to a multi-resolution surfel map. Tateno *et al.* [35] used a pre-trained database of objects to generate descriptors, but they used a KinectFusion [23] TSDF to incrementally segment regions of a reconstructed TSDF volume and match 3D descriptors directly against those of other objects in the database.

A number of approaches to object discovery exist [5, 32, 4]. Most related to ours is the work of Choudhary *et al.* [4] where they localised the camera in an online manner using discovered objects as landmarks in a pose-graph formulation similar to ours, although they used the point cloud centroid only whereas our pose-graph object landmark edges are full 6 DoF $SE(3)$ constraints provided from ICP on dense volumes. They showed that the approach improves SLAM results by detecting loop closures. However, unlike our work they use point-clouds rather than TSDFs and do not train an object detector but instead they use the unsupervised segmentation approach of Trevor *et al.* [36].

Another approach to object discovery is through dense change detection between successive mappings of the same scene [12, 19, 11]. Unlike these systems, our system is designed for online use and does not require changes to occur in a scene before objects are detected. These approaches are complementary to our proposed approach, providing supervisory signals for CNN fine-tuning, and enabling additional object database filtering mechanisms.

In RGB-only SLAM for object detection, Pillai and Leonard [26] use ORB-SLAM [21] to assist object recognition. They use a semi-dense map to produce object proposals and aggregate detection evidence across multiple views for object detection and classification. MO-SLAM by Dharmasiri *et al.* [9] focused on object discovery through duplicates. They use ORB [28] descriptors to search for sets of landmarks which can be grouped by a single rigid body transformation. This approach is similar to our relocalisation method, which uses BRISK features [18] but augmented with depth.

Very closely related to ours is work by Sünderhauf *et al.* [34], who proposed an object-oriented mapping system composed of instances using bounding box detections from a CNN and an unsupervised geometric segmentation algorithm using RGB-D data. Although the premise is closely related, there are a number of differences when compared to our system. They use a separate SLAM system, ORB-SLAM2 [22], whereas in our system the discovered object instances are tightly integrated into the SLAM system itself. We also fuse instances into separate TSDF volumes with a

foreground mask from 2D instance mask detection rather than using point cloud segments.

A number of very recent related works have also been announced. Pham *et al.* [25] fuse a TSDF of the entire scene and semantically label voxels using a CNN followed by a progressive CRF. To segment instances, instead of fusing native instance detections, they opt to cluster semantically labelled voxels in 3D. This approach, although a natural next-step from dense 3D semantic mapping, is not suitable for object-level pose graph optimisation and reconstruction as the instances are embedded within a shared TSDF. It also requires *semantic* recognition as a pre-requisite for object discovery which could prove problematic for similar or unrecognised objects in close proximity (Figure 3).

Rünz and Agapito [29], as in our method, use Mask R-CNN predictions to detect object instances. They aim to densely reconstruct and track moving instances using an ElasticFusion [38] surfel model for each object, as well as for the background static map. Although using the same prediction model, the approach and goals of these two systems differ substantially. Unlike the present work, they do not aim to reconstruct high-quality objects as pose-graph landmarks in room-scale SLAM. We on the other hand do not currently tackle dynamic scenes and assume all objects to be static during an observation. Clearly there is the long-term potential to combine these two approaches.

3. Method

Our pipeline is visualised in Figure 2. From RGB-D input, a coarse background TSDF is initialised for local tracking and occlusion handling (Section 3.3). If the pose changes sufficiently or the system appears lost, relocalisation (Section 3.4) and graph optimisation (Section 3.5) are performed to arrive at a new camera location, and the coarse TSDF is reset. In a separate thread RGB frames are processed by Mask R-CNN and the detections are filtered and matched to the existing map (Section 3.2). When no match occurs, new TSDF object instances are created, sized, and added to the map for local tracking, global graph optimisation, and relocalisation. On future frames, associated foreground detections are fused into the object's 3D 'foreground' mask alongside semantic and existence probabilities (Section 3.1).

3.1. TSDF Object Instances

Our map is composed of object instances reconstructed within separate TSDFs, $\mathcal{V}^o$, each with a pose defined by a transformation, $\mathbf{T}_{WO} \in SE(3)$, which maps coordinates of a point ${}_O\mathbf{p} \in \mathbb{R}^3$ from object frame $\underrightarrow{\mathcal{F}}_O$ to coordinates ${}_W\mathbf{p} \in \mathbb{R}^3$ in World frame $\underrightarrow{\mathcal{F}}_W$. For convenience of notation, homogeneous coordinates are assumed where appropriate (e.g. in transformations), however when explicitly required they are denoted with italics, ${}_O\boldsymbol{p} = [{}_O\mathbf{p}^\intercal, 1]^\intercal$. Object instance frames have an origin at the centre of the volume and are sized cubically with an edge-length, s_o.

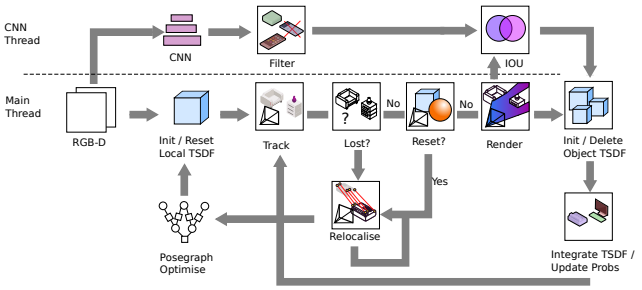

Figure 2: Overview of the Fusion++ system.

Initialisation and resizing: Detections not matched by the procedure described in 3.2 are used to initialize an appropriately sized and positioned instance TSDF. In the k^{th} frame each detection i produces a binary mask M_i^k. We project all the masked image coordinates $\mathbf{u} = (u_1, u_2)$ into $\underrightarrow{\mathcal{F}}_W$ using the depth map $D_k(\mathbf{u})$,

$$ {}_W\mathbf{p} = \tilde{\mathbf{T}}_{WC}^k \mathbf{K}^{-1} D_k(\mathbf{u}) \boldsymbol{u} \,, \tag{1} $$

where $\mathbf{K}$ denotes the 3×3 intrinsic camera matrix, $\tilde{\mathbf{T}}_{WC}^k \in SE(3)$ the camera pose estimate.

To robustly size the TSDF in the presence of masks which can occasionally include far-away background surfaces, we do not directly accept the maximum and minimum of this point cloud. Instead we use the 10^{th} and 90^{th} percentiles of this point cloud (separately for each axis) to define points $\mathbf{p}_{10}$ and $\mathbf{p}_{90}$ respectively, which are used to calculate the volume centre $\mathbf{p}_o = \frac{\mathbf{p}_{90}+\mathbf{p}_{10}}{2}$ and volume size $s_o = m\|(\mathbf{p}_{90} - \mathbf{p}_{10})\|_\infty$. We use an m of 1.5 to account for erosion and provide additional padding.

Each instance TSDF has an initial fixed resolution along a given axis of r_o, which we choose to be 64, and s_o is used to calculate the physical size of a voxel $v_o = \frac{s_o}{r_o}$. Therefore, small objects will be reconstructed with fine details and large objects more coarsely, making the map as useful as possible for a given memory footprint.

During operation matched objects may need to be resized as new detections include additional areas. To do this, the point cloud of the current mask described above is combined with a similarly eroded point cloud generated from the current TSDF reconstruction. The 3D volume encompassing them both is used to calculate the new volume centre and size as before. To avoid aliasing when re-sizing, we translate the volume centre by discrete multiples of v_o, and maintain the same v_o but increase r_o, while maintaining an even parity. We also limit the maximum voxel resolution to 128, by re-initialising the volume as though new if $r_o > 128$, and limit the maximum object size to be 3m.

Before initialising an instance we require the volume centre to be within 5m of the camera, and a 3D axis-aligned bounding box Intersection over Union (IoU) < 0.5 with any other volume already in the map. When an object centre is moved, the pose-graph node and associated measurements are also updated as described in Section 3.5.

Integration: For integrating surface measurements from a depth map D^k into $\mathcal{V}^o$ we take an approach similar to Newcombe *et al.* [23][1]. $\mathcal{V}^o$ stores at each discrete voxel location $\mathbf{v} = (v_x, v_y, v_z)$ both the current normalised truncated signed distance value $S^o_{k-1}(\mathbf{v})$ and its associated weight $W^o_{k-1}(\mathbf{v})$. If $\mathbf{v}$ projects into a camera frame pixel with a depth value less than the depth measurement plus the truncation distance, μ (here chosen as $4v_o$), then that measurement is fused into the volume in a weighted average fashion. Integration is performed on every frame where the TSDF volume is visible, when 50% of TSDF pixels are validly tracked and the ICP RMSE < 0.03 (these error metrics are described in more detail in Section 3.3). This is to maintain the reconstruction quality of instances when the camera frame may have drifted.

It is also important to note that the above surface integration is performed throughout the entire volume, regardless of whether it is a masked region or not. To store which voxels correspond to this instance's 'foreground' we also fuse instance mask detections. We view each positive or negative detection as the result of a binomial trial sampled from a latent foreground probability, $p^o(\mathbf{v} \in \text{foreground})$. We store foreground $F^o_{k-1}(\mathbf{v})$ and not foreground $N^o_{k-1}(\mathbf{v})$ detection counts as the (α, β) shape parameters in a beta distribution conjugate prior which are initialised with $(1, 1)$. When a new detection is matched and the depth measurement is within the truncation distance as above, then we also update the detection counts using the corresponding mask i:

$$F^o_k(\mathbf{v}) = F^o_{k-1}(\mathbf{v}) + M^i_k(\mathbf{K}\boldsymbol{\pi}({}_C\mathbf{p}(\mathbf{v}))), \quad (2)$$

$$N^o_k(\mathbf{v}) = N^o_{k-1}(\mathbf{v}) + (1 - M^i_k(\mathbf{K}\boldsymbol{\pi}({}_C\mathbf{p}(\mathbf{v})))), \quad (3)$$

with $\boldsymbol{\pi}([x, y, z]^\intercal) = [x/z, y/z, 1]^\intercal$ denoting the projection. Finally, to compute whether a voxel is part of the foreground we calculate the expectation,

$$E[p^o(\mathbf{v})] = \frac{F^o_{k-1}(\mathbf{v})}{F^o_{k-1}(\mathbf{v}) + N^o_{k-1}(\mathbf{v})}, \quad (4)$$

and use a decision threshold of $E[p^o(\mathbf{v})] > 0.5$. A visualisation of this is shown in Figure 3.

Raycasting: For tracking, data association, and visualisation we render depth, normals, vertices, RGB, and object indices. Within each object volume $\mathcal{V}^o$ we step along the ray with a stepsize of v^o_s (and $0.5v^o_s$ when $S^o_k(\mathbf{v}) < 0.8$, where $S^o_k(\mathbf{v})$ is the SDF normalised by μ) and search for the zero-crossing point in $S^o_k(\mathbf{v})$ where $E[p^o(\mathbf{v})] > 0.5$ (both values are trilinearly interpolated from neighbouring voxels to smooth the representation). We store the ray length of the nearest of these intersections to avoid searching past that point in another volume.

[1]Code based on https://github.com/GerhardR/kfusion.

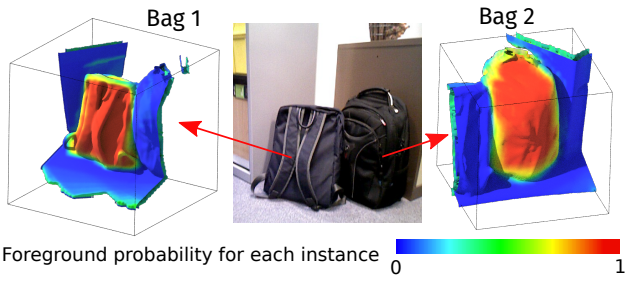

Figure 3: Object volume foreground. Note that if this value falls below 0.5 it is not rendered.

This alone results in occluding surfaces which are not part of the foreground failing to occlude the ray. If a background TSDF is available, and either no intersection with a foreground object occurs or the intersection is farther than 5cm behind the background TSDF intersection, then the background TSDF ray intersection is used instead.

Existence Probability: To prevent spurious instances from building up over time, we also model the probability of each instance's existence as $p(o)$ using the Beta distribution, in a manner identical to the foreground mask. For any frame where a predicted instance should be clearly visible (i.e. our raycasted image has more than 50^2 pixels of that instance), then if the instance has been associated to a detection its existence count e_o is incremented, and if not its non-existence count, d_o, is incremented. If $E[p(o)]$ falls below 0.1, the instance is deleted and the object node with all associated edges are removed from the pose graph (described in Section 3.5).

Semantic Labels: Each TSDF also stores a probability distribution over potential class labels l_o. Mask R-CNN provides a probability distribution $p(l_o|I_k)$ over the classes given the image, I_k. We found that the standard multiplicative Bayesian update scheme [15, 20]:

$$p(l^k_o|I_1, \ldots, I_k) = Z^{-1} p(l_o|I_k) p(l_o|I_1, \ldots, I_{k-1}), \quad (5)$$

where Z is a normalising constant, often leads to an overly confident class probability distribution, with scores unsuitable for ranking in object detection. Instead here we fuse multiple associated detections by simple averaging:

$$p(l^k_o|I_1, \ldots, I_k) = \frac{1}{k} \sum_{i=1}^{k} p(l_o|I_i), \quad (6)$$

which produces a more even class probability distribution.

3.2. Detection and Data Association

Detections from the Mask R-CNN model [13] for a given frame k contain instances i with a binary mask M_k^i and class probability distribution $p(l_i|I_k)$. A forward pass takes ∼250ms, and although our system is not real-time, this still represents a significant bottleneck and so can be performed in a parallel thread. For GPU memory efficiency, we take only the top 100 detections (scored according to the region proposal network 'object' score [27]) and filter for masks not near the image border (within 20 pixels) and where both $\max(p(l_i|I_k)) > 0.5$ and $\sum M_k^i > 50^2$.

After local tracking (Section 3.3) we use the estimated camera pose and TSDFs already initialised in the map to raycast a binary mask M_k^o for object instances o in the current view. We map each detection i to a single instance o by calculating the intersection of the two as a proportion of the *detection's* area, $a_{\text{detect}}(i,o) = \frac{\sum M_k^o \cap M_k^i}{\sum M_k^i}$ and assigning the detection to the largest intersection, $\tilde{o} = \text{argmax}_o\, a_{\text{detect}}(i,o)$, where $a_{\text{detect}}(i,\tilde{o}) > 0.2$, otherwise the detection is unassigned. For the integration step, each detection which has been mapped to the same instance is combined by taking the union of the detection masks, and the average of the class probabilities.

3.3. Layered Local Tracking

We maintain an instance-agnostic coarse background TSDF, a, to assist local frame-to-model tracking where/when there are no instances and to handle occlusions. It has a resolution of 256^3 with a voxel size of 2cm. Its initialisation point ${}_W\mathbf{p}_a = \mathbf{T}_{WC}^k[0 \quad 0 \quad 2.56]^\intercal$, is 2.56m along the z-axis in the camera frame $\underrightarrow{\mathcal{F}}_C$ to prevent wasted volume as in [37]. The volume is reset when its new initialisation point exits a spherical threshold (1.28m) around the previous volume centre, i.e. $\|{}_W\mathbf{p}_a - \mathbf{T}_{WC}^k[0 \quad 0 \quad 2.56]^\intercal\|_2 > 1.28$.

We combine the background TSDF with individual instances to raycast (Section 3.1) a 'layered' reference frame, denoted r, with vertex map, V_r, normal map, N_r, and object index map, X_r, from the previous camera pose, $\mathbf{T}_{WC_r}$, with vertices and normals defined in the world frame $\underrightarrow{\mathcal{F}}_W$. The transform to the live frame, denoted l, is estimated by aligning the live depth map, after bilateral filtering and projection to a vertex map V_l and normal map N_l with pixels $\mathbf{u}_l$, to the rendered maps with iterative closest point using projective data association and a point-to-plane error, $E_{\text{icp}}(\tilde{\mathbf{T}}_{WC_l})$, as described in [23]:

$$\boldsymbol{u}_r = \mathbf{K}\boldsymbol{\pi}(\mathbf{T}_{WC_r}^{-1}\tilde{\mathbf{T}}_{WC_l}V_l(\mathbf{u}_l)), \tag{7}$$

$$r_{\text{icp}}(\tilde{\mathbf{T}}_{WC_l}, \mathbf{u}_l) = N_r(\mathbf{u}_r)\cdot(V_r(\mathbf{u}_r) - \tilde{\mathbf{T}}_{WC_l}V_l(\mathbf{u}_l)), \tag{8}$$

$$E_{\text{icp}}(\tilde{\mathbf{T}}_{WC_l}) = \sum_{\mathbf{u}_l \in V_{\text{valid}}} r_{\text{icp}}(\tilde{\mathbf{T}}_{WC_l}, \mathbf{u}_l)^2. \tag{9}$$

Where V_{valid} includes any $\mathbf{u}_l$ with a corresponding vertex and normal, where there is a corresponding $\mathbf{u}_r$ with a valid vertex and normal, and where $N_r(\mathbf{u}_r)\cdot N_l(\mathbf{u}_l) < 0.8$ and $\|V_r(\mathbf{u}_r) - \tilde{\mathbf{T}}_{WC_l}V_l(\mathbf{u}_l)\|_2 < 0.1$m.

We minimize this non-linear least squares problem using the Gauss-Newton algorithm. We linearise $\tilde{\mathbf{T}}_{WC_l}$ about the previous estimate with the perturbation, $\boldsymbol{\zeta}$ where $\tilde{\mathbf{T}}_{WC} = \exp(\boldsymbol{\zeta})\bar{\mathbf{T}}_{WC}$. Each row of the $|V_{\text{valid}}| \times 6$ Jacobian, $\mathbf{J}_{\text{icp}}$, corresponds to the residual of a given $\mathbf{u}_l \in V_{\text{valid}}$:

$$\frac{\partial r_{\text{icp}}(\boldsymbol{\zeta}, \mathbf{u}_l)}{\partial \boldsymbol{\zeta}}|_{\boldsymbol{\zeta}=0} = -[N_r^\intercal(\mathbf{u}_r), (V_l(\mathbf{u}_l)\times N_r(\mathbf{u}_r))^\intercal]. \tag{10}$$

The Gauss-Newton iteration can then be implemented as follows (with iteration index t):

$$\boldsymbol{\zeta}^t = -(\mathbf{J}_{\text{icp}}^\intercal\mathbf{J}_{\text{icp}})^{-1}\mathbf{J}_{\text{icp}}^\intercal\mathbf{r}_{\text{icp}}, \tag{11}$$

$$\tilde{\mathbf{T}}_{WC_l}^{t+1} = \exp(\boldsymbol{\zeta}^t)\bar{\mathbf{T}}_{WC_l}^t. \tag{12}$$

The 6×6 Hessian approximation, $\mathbf{J}_{\text{icp}}^\intercal\mathbf{J}_{\text{icp}}$, and 6×1 error Jacobian, $\mathbf{J}_{\text{icp}}^\intercal\mathbf{r}_{\text{icp}}$, are reduced in parallel on the GPU and solved on the CPU using SVD and back substitution. We use a three-level coarse-to-fine pyramid scheme with 5 Gauss-Newton iterations per level.

We perform an additional reduction on the GPU to produce the same system of equations partitioned into pixels, $\mathbf{u}_l$, associated to each instance in $X_r(\mathbf{u}_r)$ for pose-graph optimisation and to produce per-instance error metrics. The error metrics are the ICP RMSE, $(|V_{\text{valid}}|^{-1}E_{\text{icp}}(\tilde{\mathbf{T}}_{WC_l}))^{\frac{1}{2}}$, and the proportion of validly tracked pixels $\frac{|V_{\text{valid}}|}{|V_l|}$. These are used for instance integration and to check whether local tracking is lost. We consider local tracking to be lost when the total ICP RMSE is greater than 0.05m or when at least 10% of the image consists of instance TSDFs and less than half of the pixels are validly tracked, in which case we enter relocalisation mode.

3.4. Relocalisation

If the system is lost or we reset the coarse TSDF, we perform relocalisation to align the current frame to the current set of instances (if there are any). We found direct dense ICP methods using only the volume reconstructions did not produce accurate results for wide baseline relocalisation as they are sensitive to the initial pose and small objects were often ambiguous without texture constraints. Although alternative dense methods may also prove useful here, we took the approach of using snapshots of sparse BRISK features[2] (with a detection threshold of 10) projected to 3D using the depth map. For a given detection of an object if there is no existing snapshot of the object within 15° view

[2] BRISK v.2 with homogeneous Harris scale space corner detection on only the highest image resolution.

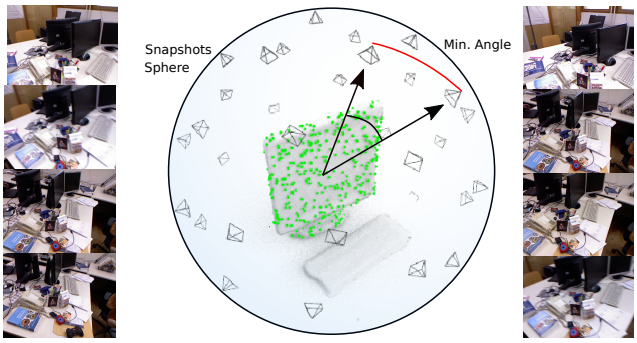

Figure 4: Re-localisation snapshots around an instance.

angle difference, we then add a new snapshot of the object from that pose (see Figure 4).

To re-localise we perform 3D-3D RANSAC against each instance where the dot product with the predicted class distribution is greater than 0.6. We use OpenGV [16] with a minimum of 5 inlier features (within 2cm) to match each object individually. If we find one or more matching objects in the scene, we run a final 3D-3D RANSAC on every point in the scene (from all objects and the background jointly) with a minimum of 50 inlier features (within 5cm) to arrive at a final camera pose. This pose is used to render a new reference image of the map to produce the constraints required for the pose graph optimisation described below.

3.5. Object-Level Pose Graph

Our pose-graph formulation is similar to that of [30]. For every frame with a Mask R-CNN detection (including coarse TSDF resets), we add a new camera pose node to our graph. When a new instance, index o, is initialised, a corresponding landmark node is added to the graph, defined by the coordinate frame attached to the centre of the object's volume, $\mathbf{p}_o$. The first camera pose node is fixed and defined to be the origin of the world frame, $\underrightarrow{\mathcal{F}}_W$. Each node consists of a full $SE(3)$ transformation from object to World, $\mathbf{T}_{WO}$, or camera to world, $\mathbf{T}_{WC}$, and the measurements are $SE(3)$ relative pose constraints between nodes.

Each relative measurement is derived by employing only the ICP error terms which correspond to the pixels of the specific object o (for object-camera constraints), or the instance-agnostic background a (for camera-camera constraints). To ensure that the measurement coincides with the minimum of the partitioned set's quadratically approximated error function, an additional Gauss-Newton step is performed using the partitioned $\mathbf{J}^o_{\text{icp}}$ (see Section 3.3) to produce 'virtual' relative pose measurements $\tilde{\mathbf{T}}'^a_{C_{k-1}C_k}$, between camera nodes, and $\tilde{\mathbf{T}}'^o_{OC_k}$, between camera and landmark objects. The resulting measurement errors for the graph factors are:

$$\mathbf{e}_{\text{cc}}(\mathbf{T}_{C_{k-1}W}, \mathbf{T}_{WC_k}) = \log((\tilde{\mathbf{T}}'^a_{C_{k-1}C_k})^{-1}\mathbf{T}_{C_{k-1}W}\mathbf{T}_{WC_k}), \tag{13}$$

$$\mathbf{e}_{\text{oc}}(\mathbf{T}^o_{OW}, \mathbf{T}_{WC_k}) = \log((\tilde{\mathbf{T}}'^o_{OC_k})^{-1}\mathbf{T}^o_{OW}\mathbf{T}_{WC_k}). \tag{14}$$

For every relative measurement, we approximate the inverse measurement covariance by $\mathbf{\Sigma}^{-1} = \mathbf{J}^{o\intercal}_{\text{icp}}\mathbf{J}^o_{\text{icp}}$. However, since the way perturbations are modelled differs between the ICP algorithm and the employed pose graph optimiser we need to transform the covariance by considering the relation between the local perturbations. The graph optimiser models perturbations $\boldsymbol{\zeta}_{\text{pg}}$ to relative pose measurements via $\tilde{\mathbf{T}}'^o_{O'C_k} = \tilde{\mathbf{T}}'^o_{OC_k}\exp(\boldsymbol{\zeta}_{\text{pg}})$ (equivalently for $\tilde{\mathbf{T}}'^a_{C_{k-1}C_k}$). To ensure our information matrix properly corresponds to perturbations $\boldsymbol{\zeta}_{\text{pg}}$, it is necessary to convert $\mathbf{J}_{\text{icp}}$. As can be seen in Eq. 12, $\mathbf{J}_{\text{icp}}$ is with respect to perturbations applied via $\tilde{\mathbf{T}}_{W'C_k} = \exp(\boldsymbol{\zeta}_{\text{icp}})\tilde{\mathbf{T}}_{WC_k}$. The relation between $\boldsymbol{\zeta}_{\text{icp}}$ and $\boldsymbol{\zeta}_{\text{pg}}$ is:

$$\exp(\boldsymbol{\zeta}_{\text{icp}})\mathbf{T}_{WC_k} = \mathbf{T}^o_{WO}\tilde{\mathbf{T}}'^o_{OC_k}\exp(\boldsymbol{\zeta}_{\text{pg}}), \tag{15}$$

$$\boldsymbol{\zeta}_{\text{icp}} = \log(\mathbf{T}_{WC_k}\exp(\boldsymbol{\zeta}_{\text{pg}})\mathbf{T}^{-1}_{WC_k}) = \text{Adj}_{\mathbf{T}_{WC_k}}\boldsymbol{\zeta}_{\text{pg}}, \tag{16}$$

$$\mathbf{J}_{\text{pg}} = \frac{\partial\boldsymbol{\zeta}_{\text{icp}}}{\partial\boldsymbol{\zeta}_{\text{pg}}} = \text{Adj}_{\mathbf{T}_{WC_k}}, \tag{17}$$

where $\text{Adj}_{\mathbf{T}_{WC_k}}$ is the Adjoint of $\mathbf{T}_{WC_k}$ such that $\exp(\text{Adj}_{\mathbf{T}_{WC_k}}\boldsymbol{\zeta}_{\text{pg}}) = \mathbf{T}_{WC_k}\exp(\boldsymbol{\zeta}_{\text{pg}})\mathbf{T}^{-1}_{WC_k}$ as described in [10]. The derivation for camera nodes results in the same transformation and the new information matrix therefore becomes,

$$\mathbf{H}_{\text{pg}} = \mathbf{J}^{\intercal}_{\text{pg}}(\mathbf{J}^{o\intercal}_{\text{icp}}\mathbf{J}^o_{\text{icp}})\mathbf{J}_{\text{pg}}. \tag{18}$$

The final error to be minimised in the pose graph is the sum over all the edges from the camera to objects, $\mathcal{O}$, and camera to camera, $\mathcal{C}$, given their state, the measurement, and the information matrix,

$$E_{\text{pg}} = \sum_{\text{cc}\in\mathcal{C}} L_\sigma(\mathbf{e}^{\intercal}_{\text{cc}}\mathbf{H}_{\text{pg}}\mathbf{e}_{\text{cc}}) + \sum_{\text{oc}\in\mathcal{O}} L_\sigma(\mathbf{e}^{\intercal}_{\text{oc}}\mathbf{H}_{\text{pg}}, \mathbf{e}_{\text{oc}}), \tag{19}$$

where L_σ denotes a robust Huber kernel. We solve this graph in the g2o [17] framework using sparse Cholesky decomposition and Levenberg-Marquart. After optimisation we update the pose of the instance TSDFs and the camera before initialising the new coarse TSDF to that pose and continuing with local tracking.

As described in Section 3.1, when a landmark is resized, its centre, $\mathbf{p}_o$, can also be adjusted from $\underrightarrow{\mathcal{F}}_O$ to a new frame $\underrightarrow{\mathcal{F}}_{O'}$ via the transform $\mathbf{T}_{O'O}$. In this case we also transform the corresponding node variable, $\mathbf{T}^o_{WO'} = \mathbf{T}^o_{WO}\mathbf{T}^{-1}_{OO}$, as well as the measurement for every edge connected to that node, $\tilde{\mathbf{T}}'^o_{O'C} = \mathbf{T}_{O'O}\tilde{\mathbf{T}}'^o_{OC}$.

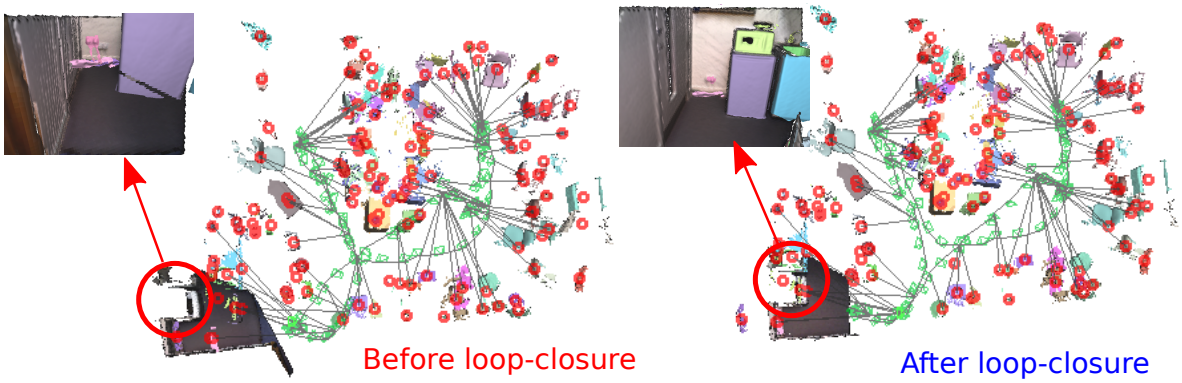

Figure 5: Comparison of office sequence trajectory before loop-closure (left) and after loop-closure (right).

4. Experiments

We evaluate the performance and memory usage of our system on a Linux system with an Intel Core i7-5820K CPU at 3.30GHz, and an nVidia GeForce GTX1080 Ti GPU with 11.175GB of memory. Our core pipeline is implemented in Python and uses Tensorflow for instance predictions, and Python wrappers around other core components which are developed in C++ and/or CUDA, such as KFusion, g2o, BRISK, and OpenGV. Our input is standard 640×480 resolution RGB-D video. To allow for reproducibility, instead of running an asynchronous CNN thread we here perform predictions synchronously every 30 frames.

Our Mask-RCNN uses the ResNet-101 base model [14] (up to the conv4_x block) and is finetuned from the publicly available `tensorpack` implementation and weights [40].[3] For finetuning on indoor scenes we use the NYUv2 dataset. We lock the ResNet-101 weights from the COCO pretraining and fine-tune the remaining layers. As the COCO dataset consists of 80 classes we re-size and reinitialise the class-specific upper layers of Mask R-CNN and Faster R-CNN. We train using stochastic gradient descent with momentum of 0.9 for 30 epochs with a learning rate of 0.001.

4.1. Loop Closure and Map Consistency

To evaluate the performance of our system while repeatedly viewing a scene of instances we captured a 3,685 frame sequence of an indoor office scene. We tailored this sequence to evaluate the consistency of our map in the presence of poorly constrained (planar floor) geometry and ICP drift, after which we loop over the same scene again. The pose-graph and loop closure is shown in Figure 5, it can be seen that despite the accumulated drift, the system relocalises and corrects the pose graph, this allows the previously reconstructed objects to be correctly associated in future frames. On the entirety of the trajectory our system reconstructed 105 landmark object instances, however, it must be noted that despite our filtering mechanisms, a build up of noisy partially reconstructed sub-objects still occurs.

[3] http://models.tensorpack.com

4.2. Reconstruction Quality

To evaluate the reconstruction quality we use objects from the YCB dataset which provides ground truth models [1] and reconstruct discovered objects from sequence 0001 of the public YCB video dataset [41]. Figure 6 shows a qualitative comparison against the ground truth. The missing portion of the cracker box was caused by an occlusion by another object, and a missed foreground detection on one of the few frames where the cracker box was unoccluded.

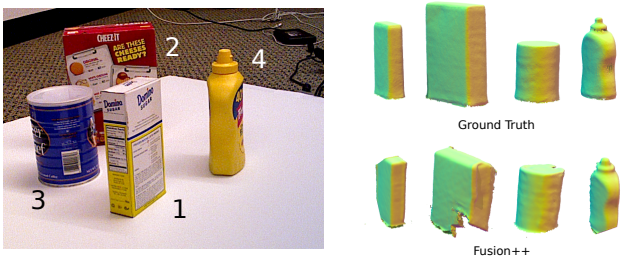

Figure 6: Reconstruction quality vs ground truth from sequence 0001 of the public YCB video dataset [41].

4.3. RGB-D SLAM Benchmark

We evaluate the trajectory error of our system against the baseline approach of simple coarse TSDF odometry, i.e. using the same coarse resetting background without instances layered on top, and without loop-closure pose graph optimisation. Table 1 shows the results. It can be seen that in all but one of the sequences evaluated our Fusion++ system improved upon the baseline approach (while providing an inventory of objects as Figure 1 visualises for the fr2_desk sequence). It is also worth noting that our system does

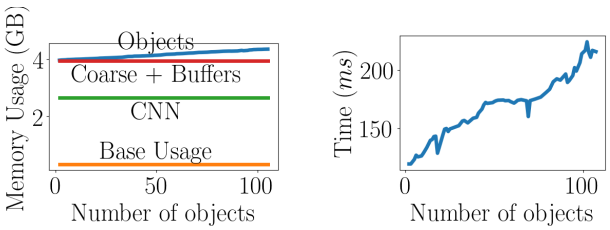

Figure 7: GPU memory usage and per-frame wall clock scaling by number of objects on the office sequence.

not achieve state-of-the-art performance on these sequences such as [38, 22], and would require additional work, such as including joint depth and photometric tracking, to become competitive. We focused on a usable object map here and leave accuracy of motion tracking for future work.

Table 1: RGB-D SLAM Benchmark ATE RMSE (m).

Sequence	TSDF Odometry	Fusion++
fr1_desk	0.066	**0.049**
fr1_desk2	**0.146**	0.153
fr1_room	0.305	**0.235**
fr2_desk	0.342	**0.114**
fr2_xyz	0.022	**0.020**
fr3_long_office	0.281	**0.108**

4.4. Memory and Run-time Analysis

Memory usage: We use the office sequence to evaluate the run-time performance and memory usage of our system. As memory usage scales cubically with the size of a TSDF, it is significantly more efficient to compose a map of many relatively small, highly detailed, volumes in dense areas of interest than to use one large one with a resolution equal to the smallest. After loading the CNN and image buffers, our remaining ∼7GB GPU memory budget (and 10 bytes per voxel) would allow a single 900^3 volume or, as here, a 256^3 background volume and up to 2.5K object volumes with dimension 64^3, 2MB. Our object volumes dynamically vary up to 128^3 and on our office sequence used 377MB for 105 objects (∼4MB/object), as shown in Figure 7. Of course, more efficient alternatives such as an octree or voxel hashing can also be used to directly eliminate wasted free-space voxels, and are also directly applicable to our approach.

Runtime performance: Our system, although not real-time, scales well with the number of objects. Excluding re-localisation on the office sequence the average frame rate was 4-8Hz (shown in Figure 7), with an average additional computational cost of 1ms per object. A more detailed breakdown of the runtime performance of different components and their scaling factors is given in Table 2.

Table 2: Run-time analysis of system components (ms) with approximate scaling performance on office sequence.

Component	Base (ms)	Scaling
Every frame		
Tracking + coarse TSDF	35	constant
Raycast all TSDFs	25	+0.5/vis. object
Object integration	15	+1.6/vis. object
On detection frames		
Mask R-CNN thread	260	constant
Detection point-cloud	10	constant
New object initialisation	-	+30/new object
Object resize+mask fuse	-	+20/vis. object
TSDF reset/re-localisation		
Relocalisation	780	+65/snapshot
Pose-graph optimisation	80	+2/object

5. Conclusions

We have shown consistent instance mapping and classification of numerous objects of previously unknown shape in real, cluttered indoor scenes. Our online and near real-time system, which is built from modules for image-based instance segmentation, TSDF fusion and tracking, and pose graph optimisation, makes a long-term map which focuses on the most important object elements of a scene with variable, object size-dependent resolution.

A number of shortcomings of the current approach remain to be addressed in future work. There is a balance to be struck between filtering detections and providing good coverage of a scene, and even with the existence probability and deletion mechanism detailed here, over time spurious detections result in a growing clutter of partial object reconstructions. More thorough object detection precision/recall evaluations as well as semantic accuracy metrics will assist in this. A learned mechanism for filtering and reconstructing these objects, such as [7] may prove useful in this regard, or combining view-based segmentation and classification with 3D methods which take advantage of object databases such as ShapeNet [2].

There is also significant scope in future to better combine information from multiple duplicate objects seen from different views to reconstruct a single better model, rather than maintaining separate TSDFs for each. Our object-oriented representation can also naturally be extended to model moving objects with individually changing poses. This attribute would be particularly useful when reasoning about dynamic applications in robotics or augmented reality.

Acknowledgements

This research was supported by Dyson Technology Ltd.

References

[1] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The YCB object and Model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015. 7

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 8

[3] S. Choi, Q. Zhou, and V. Koltun. Robust Reconstruction of Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[4] S. Choudhary, A. J. B. Trevor, H. I. Christensen, and F. Dellaert. SLAM with Object Discovery, Modeling and Mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2014. 2

[5] A. Collet, B. Xiong, C. Gurau, M. Hebert, and S. S. Srinivasa. Exploiting Domain Knowledge for Object Discovery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013. 2

[6] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*, 1996. 2

[7] A. Dai, , J. Sturm, and M. Nießner. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[8] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics (TOG)*, 36(3):24:1–24:18, 2017. 2

[9] T. Dharmasiri, V. Lui, and T. Drummond. MO-SLAM: Multi Object SLAM with Run-Time Object Discovery through Duplicates. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2016. 2

[10] E. Eade. Lie Groups for 2D and 3D Transformations. http://www.ethaneade.com/lie.pdf, 2017. 6

[11] M. Fehr, F. Furrer, D. Ivan, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena. TSDF-based Change Detection for Consistent Long-Term Dense Reconstruction and Dynamic Object Discovery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 2

[12] R. Finman, T. Whelan, and M. Kaess. Toward lifelong object segmentation from change detection in dense RGB-D maps. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*, 2013. 2

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. 1, 5

[14] 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 (CVPR)*, 2016. 7

[15] A. Hermans, G. Floros, and B. Leibe. Dense 3D semantic mapping of indoor scenes from RGB-D images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 4

[16] L. Kneip and P. Furgale. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 6

[17] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 6

[18] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariance Scalable Keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[19] L. Ma and G. Sibley. Unsupervised Dense Object Discovery, Detection, Tracking and Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 2

[20] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 4

[21] R. Mur-Artal and J. D. Tardós. ORB-SLAM: Tracking and Mapping Recognizable Features. In *Workshop on Multi View Geometry in Robotics (MVIGRO) - RSS 2014*, 2014. 2

[22] 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 (T-RO)*, 33(5):1255–1262, 2017. 2, 8

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 2, 4, 5

[24] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[25] Q. Pham, B. Hua, D. T. Nguyen, and S. Yeung. Real-time Progressive 3D Semantic Segmentation for Indoor Scenes. *arXiv preprint arXiv:1804.00257*, 2018. 3

[26] S. Pillai and J. J. Leonard. Monocular SLAM Supported Object Recognition. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015. 2

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 5

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011. 2

[29] M. Rünz and L. Agapito. MaskFusion: Real-time Recognition, Tracking and Reconstruction of Multiple Moving Objects. *arXiv preprint arXiv:1804.09194*, 2018. 3

[30] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2, 6

[31] J. Stückler and S. Behnke. Model Learning and Real-Time Tracking using Multi-Resolution Surfel Maps. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2012. 2

[32] J. Stückler and S. Behnke. Hierarchical Object Discovery and Dense Modelling From Motion Cues in RGB-D Video. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013. 2

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012. 1, 2

[34] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful Maps With Object-Oriented Semantic Mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017. 2

[35] K. Tateno, F. Tombari, and N. Navab. When 2.5D is not enough: Simultaneous Reconstruction, Segmentation and Recognition on dense SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 2

[36] A. Trevor, S. Gedikli, R. Rusu, and H. Christensen. Efficient Organized Point Cloud Segmentation with Connected Components. In *3rd Workshop on Semantic Perception Mapping and Exploration (SPME)*, 2013. 2

[37] T. Whelan, M. Kaess, H. Johannsson, M. F. Fallon, J. J. Leonard, and J. B. McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626, 2015. 5

[38] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015. 2, 3, 8

[39] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012. 2

[40] Y. Wu et al. Tensorpack. https://github.com/tensorpack/, 2016. 1, 7

[41] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv preprint arXiv:1711.00199*, 2017. 7

[42] Q. Zhou and V. Koltun. Dense Scene Reconstruction with Points of Interest. In *Proceedings of SIGGRAPH*, 2013. 2

[43] Q. Zhou, S. Miller, and V. Koltun. Elastic Fragments for Dense Scene Reconstruction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013. 2