IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 12 January 2024, accepted 6 February 2024, date of publication 21 February 2024, date of current version 3 April 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3368859*

RESEARCH ARTICLE

# RDSP-SLAM: Robust Object-Aware SLAM Based on Deep Shape Priors

**MOSES CHUKWUKA OKONKWO**[1], **JUNYOU YANG**[1], **YIZHEN SUN**[1,2], **GUANG YANG**[3], **AND FAUSTO PEDRO GARCÍA MÁRQUEZ**[4]
[1]School of Electrical Engineering, Shenyang University of Technology, Shenyang 110870, China
[2]Intelligent Robot Laboratory, Shenyang Open University, Shenyang 110020, China
[3]Department of Electronics and Information Systems Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan
[4]Ingenium Research Group, University of Castilla-La Mancha, 13001 Ciudad Real, Spain

Corresponding author: Junyou Yang (junyouyang@sut.edu.cn)

This work was supported in part by the 111 Project under Grant D23005, and in part by the Shenyang Science and Technology Plan Project under Grant 22-315-6-02.

**ABSTRACT** Object-aware systems such as Deep Shape Prior SLAM (DSP-SLAM) provide a feasible technique for creating environment sparse maps while representing scene objects as complete 3D models. Such systems provide a compelling solution for improving the intelligence of care robots and enriching the user experience in augmented reality (AR) applications. However, owing to the abrupt and unpredictable movements exhibited by users during AR engagements and the need for real-time robot responses to changes, it is imperative that sensor data is processed robustly and speedily. DSP-SLAM suffers from a low-performance speed of 10-15 fps, although it is based on ORB-SLAM2 which can run at 30 fps. This is mainly because its instance segmentation approach has an average latency of 53ms(18.86fps). To improve tracking robustness, keyframes must be processed at a fast rate. We use a state-of-the-art one-stage deep learning detector, which significantly reduces the wait time for detection-based data association during keyframe creation, and finally present Robust Deep Shape Prior SLAM (RDSP-SLAM). The results show that segmentation was performed at 20ms (50fps), while the object 3D reconstruction quality was the same as that of DSP-SLAM. RDSP-SLAM accepts RGB sequential images at 30fps and tracks them at a mean latency of 38fps.

**INDEX TERMS** Deep shape priors, instance segmentation, object-aware SLAM, object detection, object representation and reconstruction.

## I. INTRODUCTION

Visual SLAM (Simultaneous Localization and Mapping) techniques provide compelling solutions for environment awareness and self-localization for cyber-physical systems and have been applied in areas such as augmented reality (AR), robot navigation [1], robot-object interaction [2], and human-robot interaction [3]. VSLAM enables the tracking of movement in physical systems, such as robots, based on changes in the received optical sensor data.

A considerable amount of semantic information about the environment is obtained visually by humans. Visual mapping approaches based on SLAM techniques, perform detection on visual data for increased inclusion of spatial semantic information during scene mapping [4]. Improving the semantic information obtained during the mapping process has given rise to the object-level/aware/oriented SLAM research field, where individual objects in the scene are prioritized and thus (a) recognized in camera images, (b) associated with the created map (c) either used as landmarks for camera localization [5], or (d) represented as quadrics [6], [7], cuboids [7], [8], [9], voxels [10], or surfels/mesh [11], [12], [13].

Exceptionally, object-aware systems such as Deep Shape Prior SLAM (DSP-SLAM) [13] provides a feasible technique to create a sparse map of the environment on the fly, recognize

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhua Guo.

and represent found objects (even when partially visible) as complete detailed 3D mesh models while jointly estimating the position and orientation of the camera relative to the objects. With such high-level detail of the environment, designing care robots with the ability to (i) build a memory of the location of objects, (ii) perform a memory search of objects, and (iii) reason semantically about objects needed for activities of daily living can be made possible. Augmented reality (AR) and virtual reality (VR) experiences can be improved by quickly replicating the found objects in the scene.

In AR and VR applications, abrupt and unpredictable movements are exhibited by users during their engagement. In addition, real-time robot responses to situation changes and control commands necessitate robustness and speed of sensor data processing. DSP-SLAM suffers from a low performance speed of 10 - 15 frames per second (fps) [13], although it uses ORB-SLAM2 [14] as a backbone. ORB-SLAM2 has the capability of running at a real-time speed of 30 fps.

The speed limitation of DSP-SLAM is mainly due to its object detection or instance segmentation approach, that is, it uses a two-stage detector, which has an average latency of 53ms. ORB-SLAM2's tracking thread endeavors to spawn as many keyframes as possible, thereby improving the tracking robustness during challenging camera movements [15]. Object instances are segmented on the keyframes immediately after spawning. Because object detection and map-object-data-association in DSP-SLAM are embedded in the camera tracking process, a higher detection latency will consecutively increase the wait time for keyframe spawning. In our experiments, DSP-SLAM can detect objects and associate corresponding map points when sequential images are passed at a faster frame speed (above 10-15 fps) because any additional delay in frame tracking can be mitigated by retrieving the next frame, as they are all stored on the device (PC). However, in real-world applications, the camera (robot) is easily lost because the images are required to be processed as soon as they emerge. If the images are buffered, the responsiveness of the framework is reduced.

**Contribution**: To solve this problem, we used a state-of-the-art one-stage deep learning detector to obtain object 2D bounding boxes (2DBboxes) and segmentation masks, thereby significantly reducing the wait time for keyframe detection. We then present Robust Deep Shape Prior SLAM (RDSP-SLAM). In particular, the keyframe tracking latency of DSP-SLAM, which included the total time for keyframe detection, map keyframe insertion, and object data association, was significantly reduced to imbue the system with robustness and real-time operation. There was a compromise between the speed and accuracy of the detectors. DSP-SLAM's two-stage detector has mean average precision values of 42.7 $\mathrm{mAP}^{\mathrm{bbox}}$ and 38.5 $\mathrm{mAP}^{\mathrm{mask}}$, whereas RDSP-SLAM's detector has mean average precision values of 40.5 $\mathrm{mAP}^{\mathrm{bbox}}$ and 35.4 $\mathrm{mAP}^{\mathrm{mask}}$. Experimental results show that the keyframe segmentation on RDSP-SLAM was performed at an average latency of 20ms (50fps), while the object 3D model reconstruction quality is the same as that of DSP-SLAM on indoor objects.

Section II introduces related work and discusses object instance detection challenges in object-aware SLAM data association. In Section III, an overview is provided detailing the different modules of the RDSP-SLAM. The experimental results are presented and analysed in Section IV. Conclusions are presented in Section V. All entities in the map (points, keyframes, and objects) and their data properties/components are described in Section VI.

## II. RELATED WORK

Semantic object-oriented SLAM systems mainly adopt a frame-to-frame [2], [16] or keyframe-only [10], [13] approach to object detection for data association in semantic mapping. The object instance detection results are either 2D bounding boxes (2DBboxes), segmentation masks, or both. Object image semantic cues based only on 2DBboxes do not suffer from latency problems because the results can be obtained in real-time. As such, RDS-SLAM [10], EAO-SLAM [7], and SLAM-OR [9] adopt a lightweight object detection neural network to extract 2DBox detection on all frames or keyframes in real-time while simultaneously tracking the camera, creating a scene map, and representing objects as quadrics, cuboids, and voxels.

2DBboxes do not provide sufficient visual information for enriched object representation, such as visually pleasing object mesh models in maps. To address this, deep learning detectors such as Mask-RCNN [17] and Sharp-Mask [18] have been adopted for object instance segmentation. However, their incorporation introduces a challenge in real-time execution for visual-object-data association in camera tracking and mapping tasks. Rünz et al. [12] attempted to mitigate the low latency (5fps) issue of Mask-RCNN by introducing the use of geometric depth discontinuity (GDD) segmentation on RGBD frames. GDD segmentation results were fused with those obtained from Mask-RCNN to include semantic information. Although the GDD algorithm runs in real-time, it tends to provide rough edge mask results with no semantic cues. Mask-RCNN, on the other hand, provides smoother mask edges but operates at a low frequency. As a result, the fusion proposed in [12] required the implementation of a frame buffer where GDD segmentation and camera tracking was conducted on the front frame, and Mask-RCNN segmentation was performed on the back frames of the buffer. Similarly, in [11], the utilisation of a double approach based on object motion and Sharp-Mask was employed to obtain cues of objects in the mapping scene. This two-way method allows for segmentation due to motion and the inclusion of semantic cues from the neural network by fusion, even when the objects are static.

The detection pipeline, which extracts semantic cues for data association on all frames or keyframes, has been structurally placed parallel to the tracking thread [10] to improve processing speed. However, this approach necessitates increased hardware resources, specifically a larger number of CPU processing cores. Furthermore, the detection process has been placed sequentially within the tracking thread [11], [12], [13], but a lengthy detection time can negatively impact the robustness of camera tracking, even when semantic cues are only extracted from keyframes.

## III. CASE STUDY AND APPROACH

RDSP-SLAM is an object-aware slam framework with robust camera tracking/localization built on ORB-SLAM2 [15] and leverages DeepSDF [19] for the 3D reconstruction of objects in the scene. Structurally, ORB-SLAM2 involves the concurrent process of frontend camera tracking (Section III-A) and backend local mapping (Section III-B). The loop closure in ORB-SLAM2 was disabled to allow multiple keyframe views of the object. Our algorithmic framework consists of the following steps:

1) First, RGB images from monocular sensors were passed as input to the SLAM-backbone to concurrently track camera movements, spawn keyframes, and obtain sparse 3D map points of the environment based on keyframe triangulation.
2) Second, map keyframes were passed to the 2D detector module to obtain object instance bounding boxes and masks at or above real-time speed. The detection results were then evaluated to remove the background results. Only foreground object bounding boxes and masks were included during data association.
3) Next, map points belonging to the detected foreground object were selected by filtering the keyframe features through an overlap with the segmentation result.
4) The object instance label and associated map points were then used as shape priors for object mesh model generation based on DeepSDF. Afterwards, the object's 3D model shape and pose were jointly estimated and optimized as more keyframes are inserted in the mapping process.
5) Finally, the camera poses, map points, and object poses were concurrently optimised through a joint bundle adjustment process.

RDSP-SLAM has the same algorithmic architecture as DSP-SLAM and uses the same object reconstruction method. However, it is more robust to challenging camera movements as the keyframe creation latency is reduced. The keyframe creation process which included real-time detection and segmentation of keyframe objects, as proposed in this work, provided better robustness and reduces map-keyframe insertion wait time. A flowchart of our framework is illustrated in Fig. 1, and the final reconstruction results are shown in Fig. 2. RDSP-SLAM accepts RGB sequential images at 30fps and tracks them at a mean latency of 38fps. Steps 1, 2, and 3 were embedded in the tracking thread and run concurrently with steps 4 and 5 in the local mapping thread of ORB-SLAM2. In the following subsections, we discuss these steps in detail.

### A. CAMERA TRACKING

The camera tracking thread is responsible for localising the camera with every incoming frame and deciding when to insert a keyframe. Camera positions are localised based on frame inputs by finding matches to the local map and minimizing the reprojection error through motion-only bundle adjustment (BA) [14].

#### 1) INITIALIZATION

Oriented Fast and Rotated Brief (ORB) features $x_*$ were first extracted from the current frame $F_c[x_c]$ and the preceding or referenced frame $F_r[x_r]$. Using their feature descriptors $D$, matches $x_c \leftrightarrow D \leftrightarrow x_r$ were then searched. The camera position was initialized via a parallel computation of two models: (a) Homography $H_cr$, given as: $x_c = H_{cr}x_r$ and (b) Fundamental matrix $F_{cr}$, given as: $x_cF_{cr}x_r = 0$ (epipolar geometry constraint in Fig. 3). Using a model selection heuristic, the best result for $F_r$ to $F_c$ was selected; the homography model was chosen for planar scenes, whereas the fundamental matrix was selected otherwise. For successful initialization, $F_c$ and $F_r$ were automatically inserted as keyframes $K_c$ and $K_r$.

#### 2) INITIAL POSE ESTIMATES

Assuming that the last frame $F_l$ was successfully tracked, a velocity motion model is used to predict the camera pose, then map points found in $F_l$ are searched for in $F_c$.

#### 3) GLOBAL RELOCALIZATION

If tracking is lost, $F_c$ is converted into visual words [20] and the recognition database is queried to match keyframe candidates for global relocalization. Alternatively, RANSAC iterations are performed on keyframes to find the camera pose using the PnP algorithm [21].

#### 4) LOCAL MAP TRACKING

Points in map $P^i$ were then tracked by applying motion-only bundle adjustment (BA). The aim is to consistently optimize the current camera pose $R, t; R \in SO(3), t \in \mathbb{R}^3$, by minimizing the error of reprojecting local map points $P^i_l$ into feature points $x_i \in \mathbb{R}^2$ in $F_c$. The local map is a set of keyframes $k1 \in K_1$ with neighboring keyframes $k2 \in K_2$ that have covisible map point $P^i_l \in P^i_{k1\cap k2\cap F_c}$ with $F_c$. Motion-only is expressed in (1)

$$R, t = \underset{R,t}{\operatorname{argmin}} \sum_{i\in\mathbb{X}} \rho(\|x^i - \pi(RX^i + t)\|^2_\Sigma) \tag{1}$$

where $X^i \in \mathbb{R}^3$ is the world coordinate of $P^i_l$, $i \in \mathbb{X}$ is the set of matches between $x^i$ and $P^i_l$, $\rho$ is the robust Huber cost function and $\Sigma$ is the covariance matrix associated with the scale of the feature points. $\pi$ is the monocular camera intrinsic matrix or reprojection function given in (2)

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \tag{2}$$

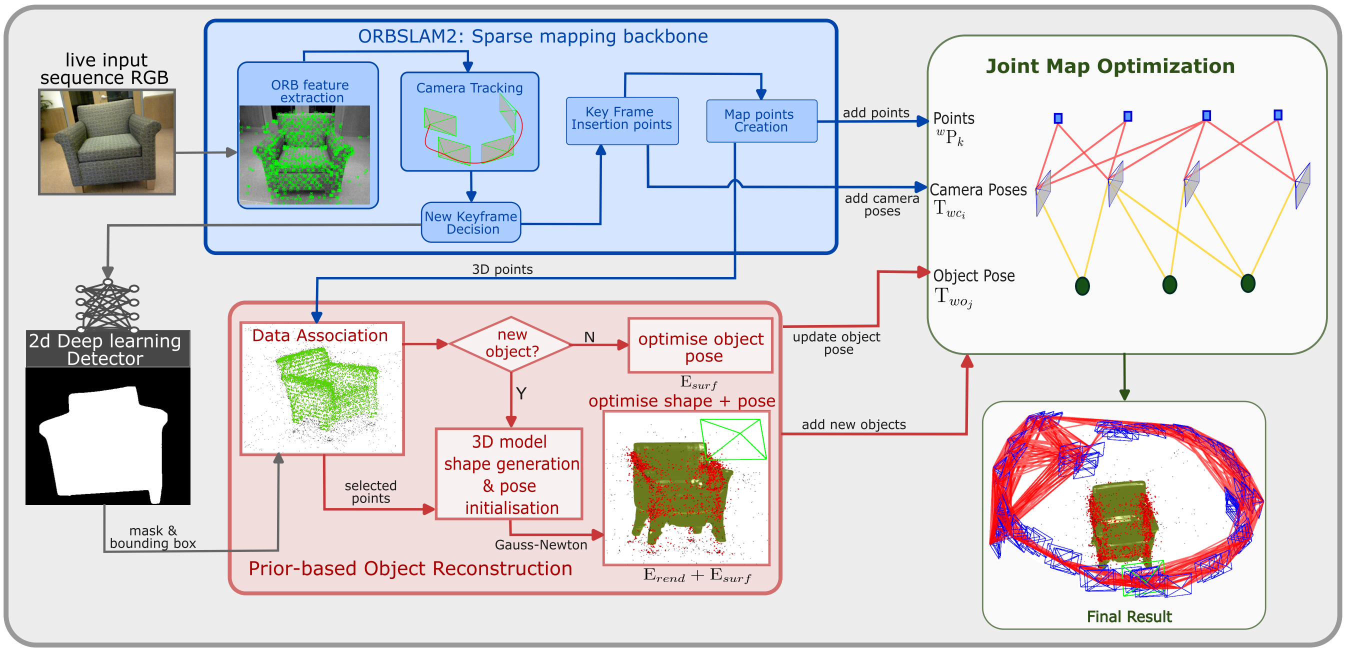

**FIGURE 1.** Object aware map creation process system overview; [Blue] ORBSLAM2 camera tracking, keyframe decision and map point creation; [Gray]Image segmentation on keyframe; [Red]Data association based on keyframe segmentation and map points, 3d model generation, pose estimation and pose optimization; [Green/Right-hand top] Joint map optimization; [Green/Right-hand bottom] Final results of the framework based on an RGB image sequence data: keyframes(blue), Covisibility graph(red line-edges), and object model.

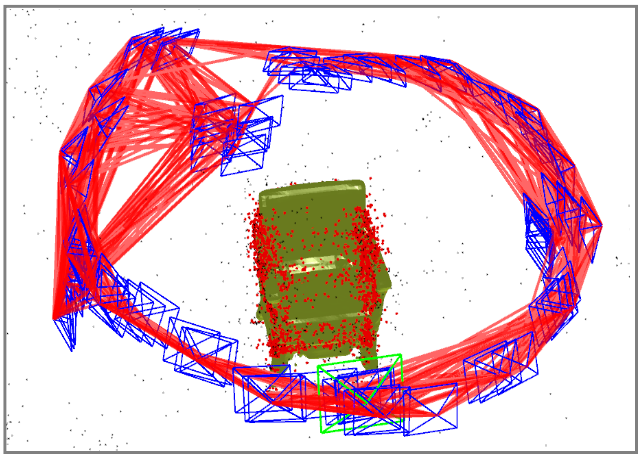

**FIGURE 2.** Sparse Map and Dense Object Reconstruction from RDSP-SLAM: keyframes(Blue boxes); Covisibility graph(edges lines in red connecting keyframes which share visible points); Object point-cloud(Red map points); Object Reconstruction(Green mesh in the center).

where $(f_x, f_y)$ is the focal length and $(c_x, c_y)$ are the principal points of the camera obtained from calibration.

### 5) NEW KEYFRAME DECISION AND CREATION

Decisions on new keyframes are made as often as possible to improve the tracking robustness during challenging camera movements, and then culled to reduce redundancy in the mapping process. $F_c$ is spawned as a keyframe if (a) a maximum number of frames (i.e., image/camera fps) or more have passed since the last keyframe insertion or global relocalization, (b) local mapping is idle, or (c) the current

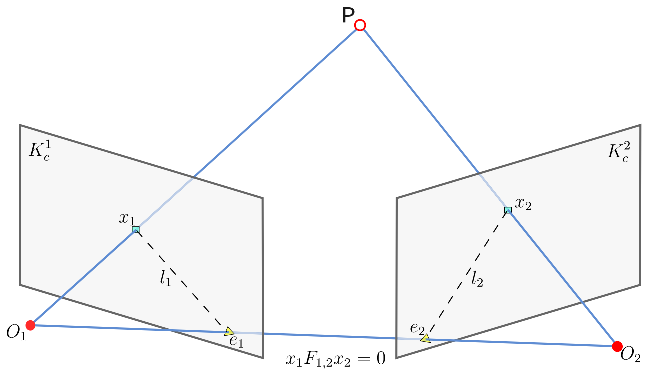

**FIGURE 3.** Epipolar Geometry constraint: $x_1$ and $x_2$ are feature points in connected keyframes $K_c^1$ and $K_c^2$ with camera centers $O_1$ and $O_2$, and covisible map point $P$.

frame tracks at least 50 and less than 90% points of $K_{ref}$. If these requirements are satisfied, a new keyframe $K_c$ is created. Creation involved storing the camera pose ${}^{w}T_c$ of $F_c$, viewable map point $P^i$, and computing its visual word description. With our proposed detection model, category-specific object segmentation mask $\mathcal{M}$ and 2d bounding box $\mathcal{B}$ of the created $K_c$ were concurrently obtained and passed to the data association module, at or above real-time speed, thus overcoming the pre-stated constraints.

### 6) DATA ASSOCIATION

Each detection on the keyframe is associated with the nearest object, $O_i$, in the map. The total number of objects $i$ to be initialized depends on the number of detections in $K_c$.

**Algorithm 1** Keyframe Decision, Creation, and Data Association

```
Data: $F_c$
Result: $O_i$
if number of frames tracked since $K_r$ > camera fps OR
  local mapping is idle OR (for $P^i$ in $F_c$, $i < 50$ OR
  $i < 0.9 * n$ (for $P_n$ in $K_r$)) then
    Create keyframe $K_c$ by converting $F_c$ to visual
      words saving its pose ${}^wT_c$ and visible $P^i$;
    $\mathcal{B}, \mathcal{M} \leftarrow$ Get object detections of $K_c$;
    Delete background object results based on the
      area of $\mathcal{B}$;
    $x_o^i \leftarrow$ Get feature points in $\mathcal{M}$ based on $K_c$;
    $P_o^i \leftarrow$ Get map point matches based on $x_o^i$ and $D_i$;
    for $P$ in $P_o^i$ do
        if $P$ is wrongly triangulated and marked as an
          outlier by the mapping process then
            Delete $P$;
        else
            Initialize $P$ with a unique in-map object
              ID $O_{id}$;
        end
    end
    $O_i \leftarrow$ Collect $P_o^i$ with same $O_{id}$ and save to
      corresponding map object;
end
```

However, only one foreground object is to be reconstructed. As detection results were obtained from keyframes in real-time, each detected object instance $I$ consisted of 2DBboxes $\mathcal{B}$, 2D masks $\mathcal{M}$, and in-object feature points $x_o^i$. We chose the foreground object based on the largest area of $\mathcal{B}$ in all $I$. All $x_o^i$ found exclusively in $\mathcal{M}$ and $K_c$ were associated with object $O_i$ in the map. By searching the keypoint to map point BRIEF descriptor correspondence $x_o^i \rightarrow D_i \rightarrow P_o^i$, map points $P_o^i$ belonging to $O_i$ were then obtained. Both $x_o^i$ and $P_o^i$ were used later in the shape reconstruction stage (Section III-B5). If a detection is not associated with a pre-existing object, it is initialized as a new object (but not reconstructed). Combined with the new keyframe decision and creation (Section III-A5), the data association is described in detail in Algorithm 1.

## B. LOCAL MAPPING

The local mapping thread manages the insertion of new keyframes. It refines keyframe poses, map point positions, and object pose by iteratively performing a local BA. New map points were created in this thread, and redundant keyframes and inconsistent map points were removed. 3D model generation through category-specific object shape code and pose optimization, together with joint map optimization, were fully integrated in the local mapping process.

### 1) KEYFRAME INSERTION

For each new keyframe insertion, the covisibility graph is updated. The covisibility graph contains a representation of keyframes as nodes and connections of weighted edge lines between keyframes that share visible map points (Fig. 2). A visual word representation was computed for each inserted keyframe.

### 2) NEW POINTS CREATION

For keyframes connected to $K_c$ in the covisibility graph, their ORB features were matched, removing feature points that do not satisfy epipolar constraints [22] (Fig. 3). New map points were then created by triangulating [22] feature points found in $K_c$.

### 3) REDUNDANCY REMOVAL

Map points that had not been tracked or wrongly triangulated in the first three keyframes were removed. Subsequently, keyframes with redundant view information were removed to reduce bundle adjustment complexity and computational cost.

### 4) LOCAL BUNDLE ADJUSTMENT

Given a local set of keyframes $K_L$, which has a set of covisible local points $P_L$, and other keyframes $K_F$ not in $K_L$, which also observe $P_L$, a nonlinear optimization is implemented to reduce the difference (cost function) of the feature points in $K_L$ and the back reprojection of $P_L$ to $K_L$. Keyframes $K_F$ contributes to the cost function but remains fixed in the optimization. If $X_k$ is the set of matches between set points $P_L$ and keypoints in keyframe $k$, then the optimization problem is given as (3)

$$\{X^i, R_l, t_l | i \in P_L, l \in K_L\} = \underset{X^i, R_l, t_l}{\operatorname{argmin}} \sum_{k \in K_L \cup K_F} \sum_{j \in \mathbb{X}_k} \rho(E_{kj})$$
$$E_{kj} = \|x^j - \pi(R_k X^j + t_k)\|^2_{\Sigma} \tag{3}$$

where $E_{kj}$ is the camera-to-map-point error term, $x^j$ are keypoints in the monocular keyframe, $X^j$ are map points with matched feature points in keyframe $k$. $R_k$ and $t_k$ are the rotation and translation of the keyframe. The local BA process was extended to include a joint optimization of the camera pose, map point, and object pose, which will be discussed in Section III-B6.

### 5) OBJECT RECONSTRUCTION AND OPTIMIZATION FROM SHAPE PRIORS

Object 3D reconstruction can be implemented by taking advantage of their category-specific-shape-embeddings as priors [2], [19], [23]. Accordingly, DeepSDF [19] regresses the continuous signed distance function (SDF) representation of class-specific object shapes. From prior shape embedded codes (or latent vector codes), watertight 3D models of an object can be obtained through regression even with incomplete (partial) or noisy 3D point input of the object.

Subsiquently, generated 3D models can be rendered using Marching Cubes [24] and visualized through ray-casting or rasterization.

### *a: RECONSTRUCTION*

First, from a map object instance $O_i = [\mathcal{B}, \mathcal{M}, P_o^i, {}^cT_{o,0}]$ obtained from the data association process, ${}^cT_o$ was initialized by calculating the principal component analysis (PCA) of $P_o^i$. The values of the dense-shape-embedded-code vector $z \in \mathbb{R}^{64}$ were initially set to zero. Then, by accounting for $P_o^i$ during the SDF calculation, an initial shape code closely fitting the actual object in the scene is acquired [13]. The SDF value $s^i$ is determined based on (4)

$$s_i = G({}^oT_c\pi_{-1}(u, P_o^i), z),$$
$$o_i = \begin{cases} 1 & s_i < -\sigma \\ -\dfrac{s_i}{2\sigma} & |s_i| < -\sigma \\ 0 & s_i > -\sigma \end{cases} \tag{4}$$

where $\sigma$ is the smoothness cutoff transition threshold value, which is set to 0.01, and $u$ is the pixel coordinate of $P_o^i$. The occupancy probability is denoted as $o_i$ and is used to determine in a piecewise manner whether a point is occupied by the object or belongs to free space.

### *b: SHAPE AND POSE OPTIMIZATION*

Using two terms, the optimization process aims to minimize the surface reconstruction inconsistency $E_{surf}$ and depth rendering loss $E_{rend}$ of the previously acquired shape code vector [13]. The surface consistency term $E_{surf}$ is formulated from (4), and thus we obtain (5).

$$E_{surf} = \frac{1}{\Omega_s} \sum_{u \in \Omega_s} G^2({}^oT_c\pi_{-1}(u, P_o^i), z) \tag{5}$$

where $\Omega_s$ denotes the pixel coordinates of $u$. Additionally, $E_{rend}$ in (6) provides a point-to-point map point $P_o^i$ pairwise depth $d_u$ constraint, restricting the object shape from growing outside the segmentation mask silhouette.

$$E_{rend} = \frac{1}{\Omega_r} \sum_{u \in \Omega_r} (d_u - \hat{d}_u)^2 \tag{6}$$

where, $\Omega_r = \Omega_s \cup \Omega_{\mathcal{B}}$ is the union of surface pixels in $\mathcal{B}$ and $\mathcal{M}$. $d_u$ and $\hat{d}_u$ are the depth and expected depth values determined by sampling the back-propagation of $u$ to $P_o^i$ using $\pi$. That is for each pixel $u$, a back-projected ray ${}^c_ix = o + d_i\pi^{-1}u_i$ can be obtained, where $\mathbf{o}$ is the camera optical center and $d_i$ is the depth value in camera coordinate space. Then in accordance with [13], [25], $Q$ discrete depth values $d_i = d_{min} + (i-1)\Delta d$ of range $[d_{min}, d_{max}]$, and $\Delta d = (d_{max} - d_{min})/(Q-1)$ are sampled, the expected depth value can be calculated as in (7)

$$\hat{d}_u = \sum_{i+1}^{Q+1} \phi_i d_i. \tag{7}$$

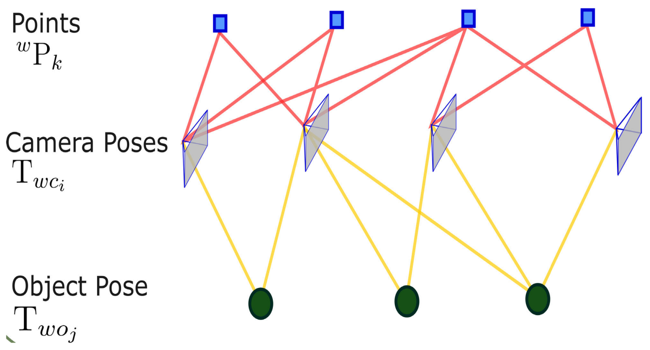

**FIGURE 4. Joint Factor Graph of Camera-to-Object and Camera-to-Map point pose.**

$\phi_i$ accounts for the probability that the ray either terminates at $d_i$ or does not intersect with the object, and is expressed as (8)

$$\phi_i = o_i \prod_{j=1}^{i-1}(1 - o_i), i = 1, \ldots, Q,$$
$$\phi_{Q+1} = \prod_{j=1}^{Q}(1 - o_i). \tag{8}$$

Finally, a Gauss-Newton optimization is formulated as shown in (9)

$$E = \lambda_s E_{surf} + \lambda_r Erend + \lambda_c \|z\|^2. \tag{9}$$

The shape and pose reconstruction optimizations were tuned using $\lambda_s$, $\lambda_c$, and $\lambda_r$ with values of 100, 2.5, and 0.25, respectively.

## 6) JOINT MAP OPTIMIZATION FOR OBJECT POSE

A joint factor graph consisting of feature points from the SLAM process $P = \{p_w^k\}_{k=1}^K$, object poses $O = \{{}^wT_o^j\}_{j=1}^M$, and camera poses $C = \{{}^wT_c^i\}_{i=1}^M$ is optimized in the local bundle adjustment process to maintain consistency between the camera-to-object and camera-to-point poses, as illustrated in Fig. 4. Using non-least square optimization, the problem is formulated as in (10)

$$C^*, O^*, P^* = \underset{\{C,O,P\}}{\operatorname{argmin}} \sum_{i,j} \|e_{co}({}^wT_{c_i}, {}^wT_{o_j})\|\Sigma_{i,j} + \sum_{i,k} \|e_{cp}({}^wT_{c_i}, p_w^k)\|_{\Sigma_{i,k}} \tag{10}$$

where $e_{co}$ and $e_{cp}$ represent the camera-to-object and camera-to-map point measurement error terms, respectively.

## IV. EXPERIMENTS AND RESULTS

We performed experiments to evaluate the keyframe processing speed of the newly proposed framework (RDSP-SLAM) and compared it with that of DSP-SLAM. The ability of the object-aware framework to track and create keyframes at a faster rate is necessary for tracking robustness given challenging camera movements. For RDSP-SLAM,

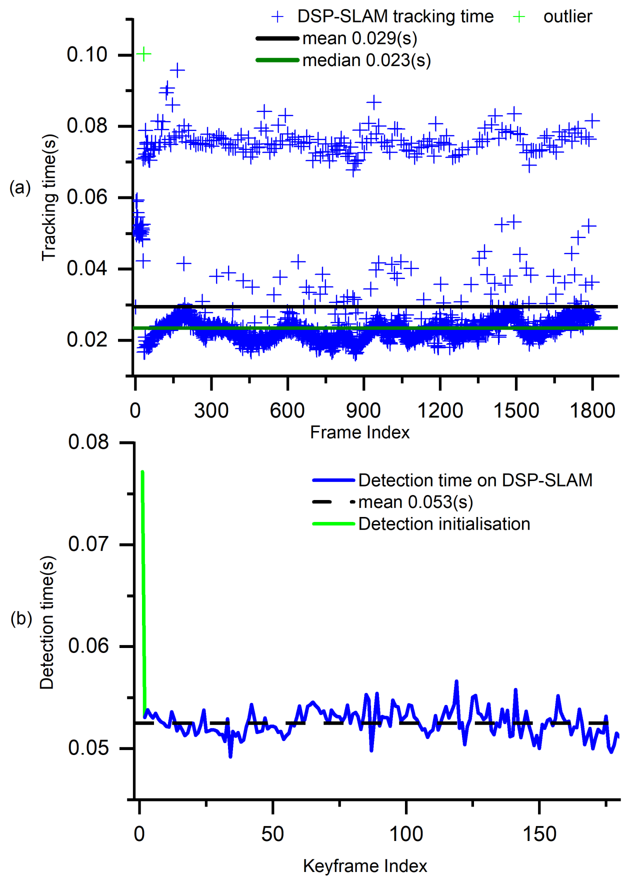

**FIGURE 5.** (a)Tracking time plot of non-keyframe and keyframes in DSP-SLAM; (b)DSP-SLAM detection time plot of all keyframes. The light green color shows the initialization time of the detector model on the first keyframe (the value is marked as an outlier).

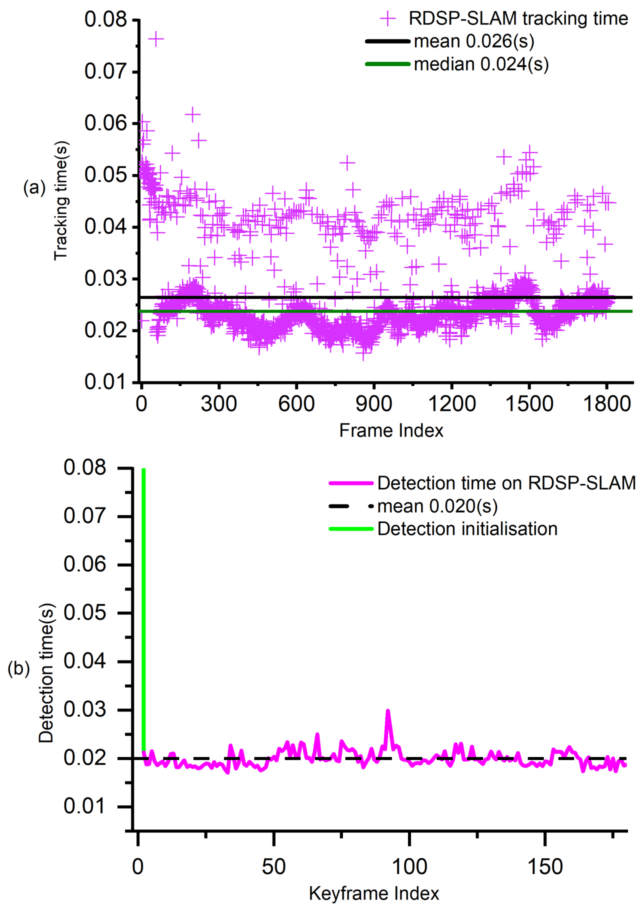

**FIGURE 6.** (a)Tracking time plot of non-keyframes and keyframes in RDSP-SLAM; (b)RDSP-SLAM detection time plot of all keyframes. Light green color shows the initialization time of the detector model on the first keyframe (the value is marked as an outlier).

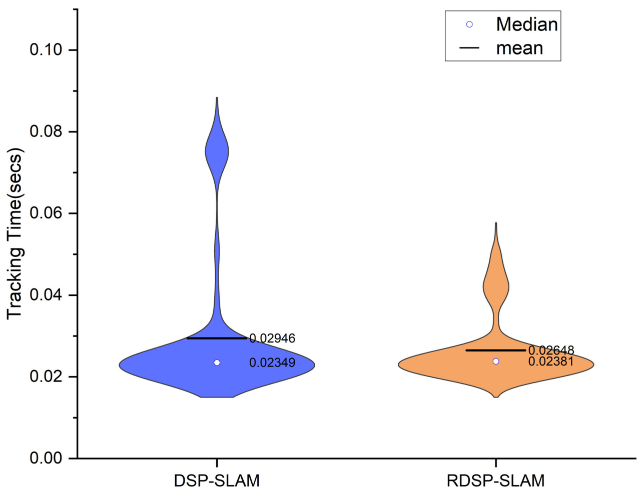

**FIGURE 7.** Density curve plot (violin plot) of tracking time on DSP-SLAM and RDSP-SLAM.

we used a one-stage neural network detector: RTMDet with 5.6 million parameters, and mean average precision (mAP) of 40.5mAP$^{box}$ and 35.4mAP$^{mask}$ [26]. DSP-SLAM uses a two-stage neural network detector: Mask R-CNN with an inference storage size of 6.1GB, and mean average precision of 42.7mAP$^{bbox}$ and 38.5mAP$^{mask}$. All detector models were pre-trained on the MS-COCO dataset. Taking the two object-aware frameworks, we input the redwood chair 09374 sequential data [27] at 30fps. The Redwood chair O9374 was used due to the large number of ORB features that can be extracted from the 2D image of the object and the feature disparity between the object and the environment. To compensate for speed, we increased the number of depth samples, object shape optimization iteration limit, and shape learning rate for RDSP-SLAM to values of 300, 10, and 2, respectively. The frontend ORB extractor aimed to detect and extract 4000 features per image. Our experiments were performed on an i7-13700KF CPU with 32G RAM and an Nvidia RTX 4070Ti GPU.

## *A. TRACKING TIME*

We evaluated the time taken to track non-keyframes and create keyframes using the two frameworks, as shown in Fig. 5(a) and 6(a). Image segmentation is part of the keyframe creation process and is necessary for data association. Hence, the detection (bounding box and mask) time plots of the two frameworks are presented in Fig. 5(b) and 6(b). The median tracking time value (which is not influenced by extreme

**TABLE 1.** Descriptive statistical values of the Tracking Time on DSP-SLAM and RDSP-SLAM; DSP-SLAM spends more time(53.2s) in tracking all frames whereas RDSP-SLAM spends less time(47.8s). Also, RDSP-SLAM has a lower tracking time standard deviation, indicating tracking time stability close to the mean(26ms).

| | No of Frames | Standard Deviation(s) | Skewness | Mean(s) | Sum(s) | Min(s) | Max(s) |
|---|---|---|---|---|---|---|---|
| DSP-SLAM [13] | 1805 | 0.017 | 2.271 | 0.030 | 53.167 | **0.016** | 0.096 |
| RDSP-SLAM | 1805 | **0.008** | **1.822** | **0.026** | **47.800** | **0.016** | **0.076** |

**TABLE 2.** The average number of inserted keyframes in the map, number of keyframes after culling, number of points, rays, and time of object reconstruction in 10 runs.

| | Avg. # of KF inserted | Avg. # of KF after culling | Avg. Point/Rays | Avg. Reconstruction time(s) / Avg. iter per run | # of runs |
|---|---|---|---|---|---|
| DSP-SLAM [13] | 181 | 106 | 6813 / 876 | 0.059 / 28.4 | 10 |
| RDSP-SLAM | 183 | 106 | 4522 / 830 | 0.486/ 8 | 10 |

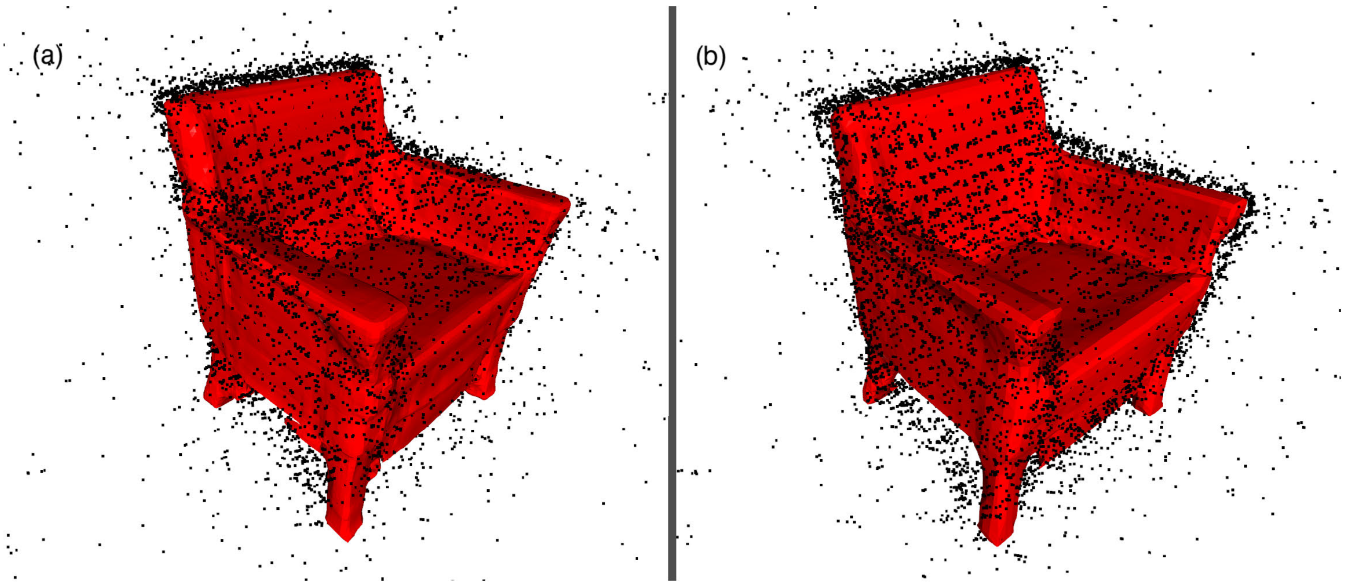

**FIGURE 8.** Object 3D reconstruction with (a) DSP-SLAM (b) RDSP-SLAM using redwood 09374 chairs sequential images passed at 30fps.

values) in the aforementioned plots sufficiently represents the average tracking time on non-keyframes, as 90 percent of the frames are non-keyframes. RDSP-SLAM and DSP-SLAM have tracking time median values of 24ms and 23ms, respectively. Conversely, the mean value changes of 26ms and 29ms for RDSP-SLAM (Fig. 6(a)) and DSP-SLAM (Fig. 5(a)), respectively, reliably show the effect of detection latency on the tracking time. Lower mean tracking time values are desirable for both real-time operation and robust tracking.

The detection time plots on all keyframes for the different frameworks are shown in Fig. 5(b) and 6(b). In RDSP-SLAM, a lower average latency of 20ms can be observed compared to DSP-SLAM, which has an average latency of 53ms. The detector models were initialized on the first keyframe creation in the two frameworks, depicted with light green lines in Fig. 5(b) and 6(b). We treated the corresponding detection time values as outliers (visible as light green dots in Fig. 5(a) in our statistical analysis).

Table 1 shows the standard deviation (SD) of RDSP-SLAM and DSP-SLAM which are 8ms and 17ms, respectively. Indicating greater consistency of detection time in RDSP-SLAM compared to DSP-SLAM, with respect to their individual mean tracking time values of 26ms and 30ms. The density curve distribution of the tracking time in the two frameworks can be seen in Fig. 7, which shows that RDSP-SLAM and DSP-SLAM are both positively skewed (tailed upward) relative to a normal distribution, with values of 1.822 and 2.271, respectively. Positive skewness means that the mean value is higher than the median value, while the skewness value shows the degree of difference. For a normal distribution, the mean and median values are the same. The positively skewed density curve distribution of the tracking time of both frameworks is because non-keyframe require less tracking time (the average tracking latency value is approximate to the median tracking value), where as keyframes require more tracking time. The tracking of keyframes introduces additional processing overhead, mainly due to keyframe creation, object instance detection and data association, which we attempt to reduce by using a state-of-the-art one-stage detector. Lower SD and skewness values are desirable for robust and stable tracking because they indicate lower overhead. Finally, a higher total tracking time of 53.167s is experienced with DSP-SLAM compared to RDSP-SLAM, 47.8s, indicating the RDSP-SLAM spends less time overall on creating new keyframes and tracking them.

### B. DATA ASSOCIATION AND OBJECT RECONSTRUCTION QUALITY

We evaluated the effect of faster frontend keyframe creation and data association on the backend object reconstruction quality when sequential data (Redwood Chair 09374) were passed into the frameworks at 30fps. From Table 2, we can observe that in 10 runs, DSP-SLAM has a higher number of included object 3D points (6813) and rays (876), whereas RDSP-SLAM includes 4522 points and 830 rays. Although more points were included with DSP-SLAM, both frameworks provide a visually sufficient and satisfying reconstruction of the object (chair), as shown in Fig. 8. Moreover, DSP-SLAM requires more tracking time, as shown in Table 1. During the experiments, any additional delay in tracking the frames could be accommodated easily by retrieving the next frame, as they were all stored in the device (PC). However, in real-world applications, the camera can easily be lost as image frames must be tracked as soon as they emerge. Notwithstanding, the larger time interval between keyframe insertions of DSP-SLAM is advantageous for better reconstruction quality because more regression iterations can occur given each incremental object 3D point data input.

## V. CONCLUSION AND FUTURE WORK

Object-aware SLAM systems provide a compelling solution for object-level awareness in real-world applications such as the enrichment of augmented reality experiences by easily replicating scene objects and robot-to-object spatial memory. The ability of the object-aware framework to track and create keyframes at a faster rate is necessary for tracking robustness given challenging camera movements are prominent in these applications. To this end, we presented RDSP-SLAM and compared its tracking latency and object reconstruction quality with those of DSP-SLAM. RDSP-SLAM has a lower mean and total tracking time of 26ms and 47.8s, respectively, which is desirable for robustness and real-time operation compared with DSP-SLAM, which has a mean and total tracking of 29ms and 53.2s. This is mainly due to RDSP-SLAM's adoption of a state-of-the-art one-stage detector running at 50fps compared with the traditional two-stage detector adopted in DSP-SLAM, which runs at 18.9fps. The use of the one-state detector with lower inference latency reduces the overall processing time of each keyframe. Both frameworks provide a visually satisfactory reconstruction of the scene object, given RGB data (Redwood-09374) input at 30fps. In future work, we will focus on removing the effect of detection initialization on the tracking thread. Additionally, a detector capable of fine-grain category-based segmentation of hollowed objects in a scene will provide close-to-ground-truth results for map-point data association and object reconstruction.

**TABLE 3.** Map data structure.

| Map Entity | Data components | |
|---|---|---|
| Map points $P^i$ | $X^i \in \mathbb{R}$, | 3D positions in world coordinates. |
| | $n_i$, | Mean unit vector of all viewing direction i.e., the rays that join the point with the optical center of the keyframes that observe it. |
| | $D_i$, | Representative ORB descriptor with minimum hamming distance with respect to all other descriptors in the key frames in which the point is visible. |
| Keyframe $K_i$ | ${}^wT_c$ | Camera poses in world coordinates, which is a combination of rotation $R \in SO(3)$ and translation $t \in R^3$. |
| | $\pi$ | Camera intrinsic. |
| | $f_x, f_y$ | Focal length. |
| | $c_x, c_y$ | Principal points. |
| | $F_*$ | Image frame. |
| | $x_*$ | ORB extracted features of the frame $F_*$; |
| Map Object $O_i$ | $K_c$ | Current keyframe on which detection is performed; |
| | $K_i$ | Keyframe in which the particular object is found; |
| | $x_o^i$ | Feature points within the detection mask; |
| | $\mathcal{M}$ | Detection mask of the foreground object in the keyframe; |
| | $\mathcal{B}$ | Detection bounding box of the foreground object in the keyframe; |
| | $O_id$ | Identification number of $O_i$ |
| | $P_o^i$ | Map points belonging to $O_i$; |

## VI. NOMENCLATURE

See Table 3.

## REFERENCES

[1] M. Kulkarni, P. Junare, M. Deshmukh, and P. P. Rege, "Visual SLAM combined with object detection for autonomous indoor navigation using Kinect v2 and ROS," in *Proc. IEEE 6th Int. Conf. Comput., Commun. Autom. (ICCCA)*, Dec. 2021, pp. 478–482.

[2] E. Sucar, K. Wada, and A. Davison, "NodeSLAM: Neural object descriptors for multi-view shape reconstruction," in *Proc. Int. Conf. 3D Vis. (3DV)*, Nov. 2020, pp. 949–958.

[3] J. Chen, B. Sun, M. Pollefeys, and H. Blum, "A 3D mixed reality interface for human–robot teaming," 2023, *arXiv:2310.02392*.

[4] W. Chen, G. Shang, A. Ji, C. Zhou, X. Wang, C. Xu, Z. Li, and K. Hu, "An overview on visual SLAM: From tradition to semantic," *Remote Sens.*, vol. 14, no. 13, p. 3010, Jun. 2022.

[5] Z. Liao, Y. Hu, J. Zhang, X. Qi, X. Zhang, and W. Wang, "SO-SLAM: Semantic object SLAM with scale proportional and symmetrical texture constraints," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4008–4015, Apr. 2022.

[6] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM," 2018, *arXiv:1804.04011*.

[7] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr, "EAO-SLAM: Monocular semi-dense object SLAM based on ensemble data association," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 4966–4973.

[8] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[9] P. Mazurek and T. Hachaj, "SLAM-OR: Simultaneous localization, mapping and object recognition using video sensors data in open environments from the sparse points cloud," *Sensors*, vol. 21, no. 14, p. 4734, Jul. 2021.

[10] X. Chen, J. Xue, J. Fang, Y. Pan, and N. Zheng, "Using detection, tracking and prediction in visual slam to achieve real-time semantic mapping of dynamic scenarios," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 666–671, Oct. 2022.

[11] M. Rünz and L. Agapito, "Co-fusion: real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4471–4478.

[12] M. Rünz, M. Buffier, and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," 2018, *arXiv:1804.09194*.

[13] J. Wang, M. Rünz, and L. Agapito, "DSP-SLAM: Object oriented SLAM with deep shape priors," 2021, *arXiv:2108.09481*.

[14] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[16] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," 2017, *arXiv:1703.06870*.

[18] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *Proc. ECCV* (Lecture Notes in Computer Science), vol. 9905, Oct. 2016, pp. 75–91.

[19] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jan. 2019, pp. 165–174.

[20] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[21] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, Feb. 2009.

[22] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision," in *Multiple View Geometry in Computer Vision*, Mar. 2004.

[23] M. Runz, K. Li, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, and R. Newcombe, "FroDO: From detections to 3D objects," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, May 2020, pp. 14708–14717.

[24] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[25] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8754–8765, Dec. 2022.

[26] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, "RTMDet: An empirical study of designing real-time object detectors," 2022, *arXiv:2212.07784*.

[27] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," 2016, *arXiv:1602.02481*.

**MOSES CHUKWUKA OKONKWO** received the B.Eng. degree in electronic information engineering from Xiangtan University, China, in 2021. He is currently pursuing the M.Sc. degree with Shenyang University of Technology, China. His current research interests include object recognition and simultaneous localization and mapping.

**JUNYOU YANG** received the degree from Harbin Institute of Technology, Harbin, China. He is currently a Distinguished Professor in Liaoning and also the first hundred-level candidate in the BaiQianWan Talents Program. He has led more than 50 research projects and has more than 200 publications in his technical field. His research interests include wind energy, special motors, and their control.

**YIZHEN SUN** received the B.S. degree in electronic information engineering and the M.S. degree in control theory and control engineering from Shenyang University, Shenyang, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the School of Electrical Engineering, Shenyang University of Technology, Shenyang. His research interests include visual/laserSLAM, perception, and autonomous navigation of various mobile robots.

**GUANG YANG** received the B.Sc. and M.Sc. degrees in electrical engineering from Shenyang University of Technology, China, in 2014 and 2017, respectively, and the Ph.D. degree from Kochi University of Technology, Japan, in 2020. He is currently an Assistant Professor with Osaka Institute of Technology. His current research interests include personal care robots, modular robots, and construction robots. He is also a member of the Robotics Society of Japan and the Society of Life Support Engineering.

**FAUSTO PEDRO GARCÍA MÁRQUEZ** is currently associated with the Ingenium Research Group, Universidad de Castilla-La Mancha, Ciudad Real, Spain.