

NeuSE: Neural SE(3)-Equivariant Embedding for Consistent Spatial Understanding with Objects

Jiahui Fu, Yilun Du, Kurran Singh, Joshua B. Tenenbaum, and John J. Leonard
 MIT CSAIL

<https://neuse-slam.github.io/neuse/>

Abstract—We present NeuSE, a novel Neural SE(3)-Equivariant Embedding for objects, and illustrate how it supports object SLAM for consistent spatial understanding with long-term scene changes. NeuSE is a set of latent object embeddings created from partial object observations. It serves as a compact point cloud surrogate for complete object models, encoding full shape information while transforming SE(3)-equivariantly in tandem with the object in the physical world. With NeuSE, relative frame transforms can be *directly* derived from inferred latent codes. Our proposed SLAM paradigm, using NeuSE for object shape and pose characterization, can operate independently or in conjunction with typical SLAM systems. It directly infers SE(3) camera pose constraints that are compatible with general SLAM pose graph optimization, while also maintaining a lightweight object-centric map that adapts to real-world changes. Our approach is evaluated on synthetic and real-world sequences featuring changed objects and shows improved localization accuracy and change-aware mapping capability, when working either standalone or jointly with a common SLAM pipeline.

I. INTRODUCTION

The ability to conduct consistent object-level reasoning is crucial for many high-level robotic tasks, especially those involving repetitive traversal in the same environment, such as household cleaning and object retrieval. In a constantly evolving world, robots are expected to accurately locate themselves and their target while keeping an updated map of the environment, ensuring that a specific “blue coffee mug” can always be retrieved regardless of its location since the last use.

Traditional Simultaneous Localization and Mapping (SLAM) approaches [2, 9, 17] see the world through a static set of low-level geometric primitives extracted from observations, making themselves less amenable to human-like reasoning about the world. In the absence of semantic information, these unordered collections of points, lines, or planes are not completely compatible with object-level interpretation, making it susceptible to false correspondence matches when faced with scene changes over time.

As the world changes and operates under the minimal unit of objects, objects serve as an intuitive source for assisting localization and an object-centric map can act as a lightweight and flexible reflection of the latest environment layout. To bridge the communication between objects and typical SLAM systems, previous works have experimented with various object representations to guide back-end optimization, ranging from pre-defined object model libraries [32, 40], semantic segmentation masks [21, 30, 22, 43], to parameterized geometry [25, 12, 44]. But they are confined to either a limited

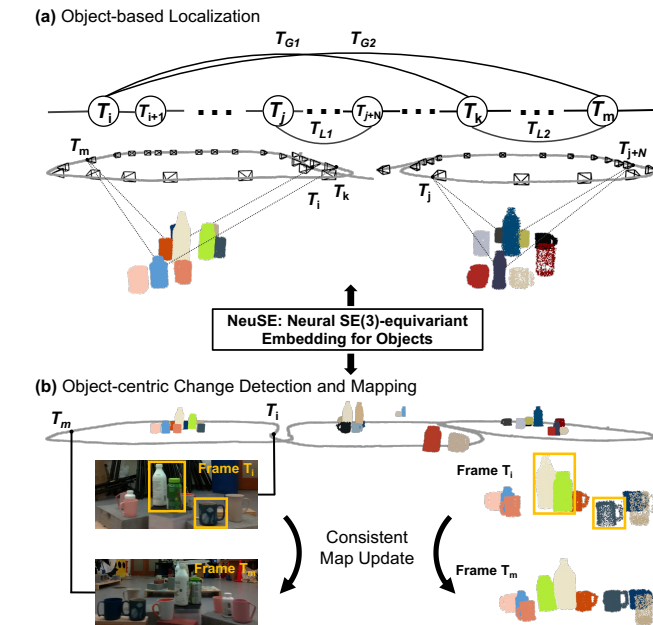


Fig. 1: **Schematic of consistent spatial understanding with NeuSE.** Object-centric map of mugs and bottles constructed from a real-world experiment is shown for illustration. (a) NeuSE acts as a compact point cloud surrogate for objects, encoding full object shapes and transforming SE(3)-equivariantly with the objects. Latent codes of bottles and mugs from different frames can be effectively associated (dashed line) for *direct* computation of inter-frame transforms, which are then added to constrain camera pose (T_i) optimization both locally (T_{L_i}) and globally (T_{G_i}). (b) The system performs change-aware object-level mapping, where changed objects (highlighted in orange) are updated alongside unchanged ones with full shape reconstructions in the object-centric map.

number of objects, or a loss of geometric details due to partial reconstruction or simplification of object shapes.

Recently, neural implicit representations have been introduced [38, 41, 49, 39] to SLAM as object or scene representations, working with probabilistic rendering loss to help constrain camera localization. However, the rendering process is parameterized as a neural network with no physical meaning, thus requiring iterative optimization with a good initialization to gradually reflect the correct SE(3) camera pose constraint embedded within the observation. This incurs extra training

and computation overhead and thus makes the integration of neural representations a cumbersome process.

In order to leverage the shape description power of neural representations while bypassing the undesirable iteration, we therefore break with the dominant “*render-optimize*” convention in previous works by explicitly imposing SE(3)-equivariance onto the vanilla representation.

Hence we introduce NeuSE, a novel category-level **Neural SE(3)-Equivariant Embedding** for objects. NeuSE learns a latent canonical point cloud from partial object observations, encoding the full object shape while transforming SE(3)-equivariantly as the object transforms in the physical world. Consequently, relative frame transforms can be *directly* computed from the corresponding latent codes of an object when it is observed in different frames. To account for pose ambiguity arising from symmetrical geometry, we further train NeuSE’s behaviors to conform to object geometric ambiguity. In this way, working with NeuSE is akin to working with the full object model, only with operations applied to a compact latent point cloud surrogate with known correspondences.

In this paper, we present NeuSE and further demonstrate how it supports object SLAM targeting spatial understanding with long-term scene inconsistencies (see Fig. 1). By using NeuSE for object shape and pose characterization, we unify the representations of major SLAM modules, e.g., data association, pose constraint derivation, etc., around one versatile latent code. Our proposed approach can either work standalone or complement common SLAM systems by *directly* inferring SE(3) camera pose constraints compatible with general SLAM pose graph optimization and maintaining a lightweight object-centric map with change-aware mapping ability (see Fig. 2).

Our main contributions are as follows: **(1)** We introduce NeuSE, a neural SE(3)-equivariant embedding for objects, encoding the full object shape and transforms SE(3)-equivariantly with the real-world object. **(2)** We propose a NeuSE-based object SLAM paradigm targeting long-term scene inconsistencies, enabling NeuSE-predicted object-level localization and change-aware mapping. **(3)** We evaluate our approach on both synthetic and real-world sequences and demonstrate improved localization performance and flexible mapping capability when working standalone or jointly with a common SLAM pipeline.

II. RELATED WORK

A. Object SLAM

SLAM++ [32] introduced object-based SLAM by incorporating camera-object constraints with objects from a predefined model database. Attempts [21, 30, 22, 43] were made to leverage semantic segmentation for instance-level dense reconstructions. Furthermore, simple parameterized geometry, e.g., ellipsoids adopted by Nicholson et al. [25] and Hosseinzadeh et al. [12] and cuboids by Yang and Scherer [44], were explored to guide the joint optimization of the object shape parameters and camera poses. For environments with moving objects, Strecke and Stueckler [37] proposed an object-level SLAM approach that utilizes local Signed Distance Function

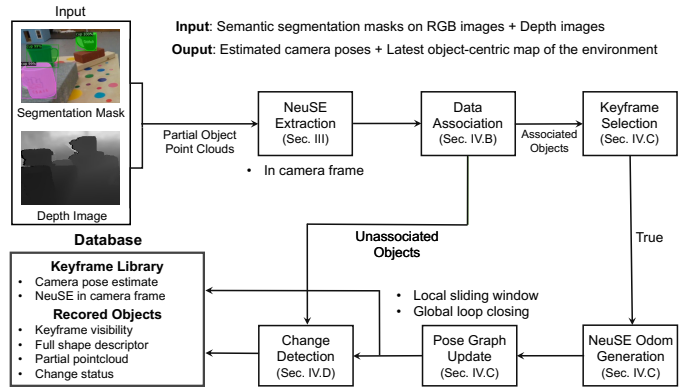


Fig. 2: **System overview.** We propose a NeuSE-based object SLAM approach targeting consistent spatial understanding with long-term scene changes.

(SDF) object volumes for tracking moving objects and performing camera localization. Recently, efforts have been made to integrate neural shape priors into the object SLAM pipeline. NodeSLAM [38] adopted a class-level optimizable object shape descriptor and used RGB-D images for joint estimation of object shapes, poses, and camera trajectory through iterative probabilistic rendering optimization. DSP-SLAM [41], on the other hand, used DeepSDF [28] for object representation and optimized the object code, camera poses, and sparse landmark points all together through a similar rendering loss in RGB, stereo, or stereo+LiDAR modalities. As the rendering process is parameterized as a neural network with no interpretable meaning, both methods require iterative optimization with a proper initialization to obtain the SE(3) transform constraint that aligns with the real-world observation. This results in added training and computational expenses, making the adoption of neural representations a complex process.

B. Neural Implicit Representations for Robotics

Neural implicit representations have emerged as a promising tool to encode the underlying 3D geometry of objects and scenes [28, 23, 26]. Different works have explored how neural implicit representations can be used in various fields, including change detection [10], localization [1, 24], SLAM [46, 5, 39, 49, 29], and manipulation [14, 13, 45, 4, 36, 20, 16, 34, 31, 18].

Notably, some works extend the original representation by integrating SO(3) or SE(3) equivariance for tasks such as reconstruction [8], point cloud registration [48, 19], and manipulation [35]. Zhu et al. [48] learned SO(3)-equivariant features to perform correspondence-free point cloud registration, while Lin et al. [19] used SE(3)-equivariant representations to obtain and refine the registration result globally and locally. Simeonov et al. [35] learned SE(3)-equivariant object representations for manipulation and estimated relative transforms through optimization. These methods target point clouds known to be associated with the same object and can suffer from performance degradation for partially overlapped point clouds [48, 19] or require iterative refinement to recover the desired relative transform [35].

In the context of SLAM, most works, other than the object-based methods listed in the previous section, utilize scene-level neural implicit representations to be jointly optimized with camera poses. iMap [39] showed that a multilayer perceptron (MLP) can serve as the scene representation for real-time RGB-D SLAM. NICE-SLAM [49], built on top of iMap, further introduced a hierarchical grid-based neural encoding, enabling RGB-D SLAM on a larger scale. In terms of monocular SLAM, recently, Nerf-SLAM [29] relied on an indirect loss for pose estimation and produced higher quality reconstructions by supervising the radiance field with depth information. These methods, like their object-based counterparts, still require undesirable iterative optimization with photometric or depth loss for localization while being hard to adapt to changes with the scene represented as one single code.

Our NeuSE-based SLAM paradigm distinguishes itself from prior SLAM works with neural representations by further explicitly imposing SE(3)-equivariance onto the vanilla neural object representations. To handle unknown data associations, in contrast to the previous works on point cloud registration or manipulation with equivariant representations, we take a step beyond to enforce shape code consistency across viewing angles. This allows partial point clouds to be matched, regardless of viewing angle differences. With additional regularization on objects with pose ambiguity, we ultimately achieve *direct* inference of SE(3) camera pose constraints from partial object representations. This eliminates the need for the computationally expensive “render-optimize” process and offers a lightweight as well as flexible solution to object SLAM problems with long-term changes.

III. CATEGORY-LEVEL NEURAL SE(3)-EQUIVARIANT EMBEDDING (NEUSE) FOR OBJECTS

We propose to represent each object in a scene by using a corresponding SE(3)-equivariant latent embedding. Precisely, given a point cloud $\mathbf{P} \in \mathbb{R}^{N \times 3}$, we represent it with a lower dimensional latent embedding (“a canonical latent point cloud”) $\mathbf{z} \in \mathbb{R}^{D \times 3}$, inferred using a neural network encoder f so that $\mathbf{z} = f(\mathbf{P})$. The underlying latent embedding is equivariant, so that for any SE(3) transform \mathbf{T} :

$$\mathbf{Tz} = f(\mathbf{TP}), \quad (1)$$

i.e. the latent embedding \mathbf{z} transforms equivariantly with respect to the point cloud \mathbf{P} .

By representing objects using this equivariant embedding, we obtain the following three benefits:

Latent Pose Constraints. The underlying latent embedding space operates under the same SE(3) action as point clouds. Thus, we may express pose constraints between matched objects directly in the latent space as opposed to the full point cloud space of objects. As the latent space is both low dimensional and canonical, pose constraints may be more efficiently computed with the closed-form solution developed by Horn [11].

Implicit Pose Representation. The object latent code implicitly captures the underlying SE(3) transform of an object. This circumvents the need to explicitly specify 6DOF poses of objects when computing pose constraints, which may not always be accessible and can be ill-defined for objects with symmetrical ambiguity.

Implicit Shape Representation. The object latent code richly encodes both the underlying shape and features of an object, which then allows for robust data association against viewing angle disparity.

To infer SE(3)-equivariant latent codes, NeuSE uses a SO(3)-equivariant encoder function [8] $f_\theta(\mathbf{P}) = \mathbf{z}$ that maps a partial object point cloud \mathbf{P} into a global latent point cloud \mathbf{z} , and a decoder function $\Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P}))$ that maps an input query point \mathbf{x} to its predicted occupancy value according to \mathbf{z} :

$$\begin{aligned} f_\theta(\mathbf{P}) = \mathbf{z} : \mathbb{R}^{n \times 3} &\rightarrow \mathbb{R}^{k \times 3} \\ \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) = \Phi(\mathbf{x}, \mathbf{z}) : \mathbb{R}^3 \times \mathbb{R}^{k \times 3} &\rightarrow [0, 1]. \end{aligned} \quad (2)$$

By feeding $\Phi(\cdot, \cdot)$ with a point cloud \mathcal{X} obtained via uniform sampling within a large bounding box centered around \mathbf{P} , the full shape point cloud \mathcal{S} of the object can be reconstructed in terms of the predicted occupancy values with $\mathcal{S} = \{\mathbf{x} | \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) > v_0, \mathbf{x} \in \mathcal{X}\}$, where v_0 is the threshold to mark whether a point location is occupied.

A. Learning SE(3)-equivariance across Viewing Angles

We construct SE(3)-equivariance separately through rotation and translation equivariance.

For rotation equivariance, as our encoder is rotation equivariant, when a point cloud is rotated by \mathbf{R} , the inferred latent code will be equivalently rotated by \mathbf{R} :

$$f_\theta(\mathbf{RP}) = \mathbf{Rz}, \mathbf{R} \in \mathbb{SO}(3). \quad (3)$$

Since \mathbf{P} is a partial observation of the complete object geometry, we treat this partial center $\bar{\mathbf{P}}$ as an initial estimate of the actual object translation so as to learn an approximately translation equivariant latent \mathbf{z} . We first infer \mathbf{z}_0 for the zero-centered partial point cloud $\mathbf{P} - \bar{\mathbf{P}}$. The final latent \mathbf{z} for point cloud \mathbf{P} is obtained by adding back the partial center $\mathbf{z} = \bar{\mathbf{P}} + \mathbf{z}_0$. Hence, to infer an SE(3)-equivariant \mathbf{z} , the final formulation of Eq. 2 is accordingly written as:

$$\begin{aligned} f_\theta(\mathbf{P} - \bar{\mathbf{P}}) = \mathbf{z}_0 : \mathbb{R}^{n \times 3} &\rightarrow \mathbb{R}^{k \times 3} \\ \mathbf{z} = \mathbf{z}_0 + \bar{\mathbf{P}}, \mathbf{z}' = \mathbf{z} - \bar{\mathbf{z}} & \\ \Phi(\mathbf{x}, \mathbf{f}_\theta(\mathbf{P})) = \Phi(\mathbf{x} - \bar{\mathbf{z}}, \mathbf{z}') : \mathbb{R}^3 \times \mathbb{R}^{k \times 3} &\rightarrow [0, 1], \end{aligned} \quad (4)$$

where $\bar{\mathbf{z}}$ is the center of \mathbf{z} . The translational equivariance on \mathbf{z} is imposed by training the center of \mathbf{z}_0 to learn the offset between $\bar{\mathbf{P}}$ and the true object center (translation). Ultimately, for the same object observed partially with camera view \mathbf{T}_1 and \mathbf{T}_2 , the SE(3)-transform $\mathbf{T}_{1,2} = (\mathbf{R}, \mathbf{t})$ between the two latent point clouds \mathbf{z}_1 and \mathbf{z}_2 , which is expected to be close to $\mathbf{T}_2^{-1}\mathbf{T}_1$, can be obtained by:

$$\mathbf{T}_{1,2} = (\mathbf{R}, \mathbf{t}) = \Psi(\mathbf{z}_1, \mathbf{z}_2), \quad (5)$$

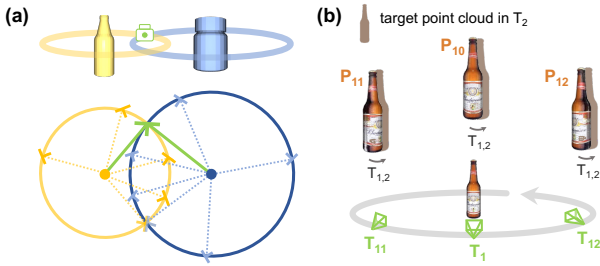


Fig. 3: **(a) Breaking pose ambiguity with covisible ambiguous objects.** Motions around a bottle’s axis of symmetry result in seemingly identical observations, making it impossible to determine inter-frame transformations. However, with two covisible bottles, the intersection (green) of their camera pose distributions (yellow and blue) for the current observation reveals the true camera pose, where inter-frame transforms can then be determined without ambiguity. **(b) Latent symmetry.** The canonicalized latent embedding should be invariant with camera motion (\mathbf{T}_{1i}) around the object’s axis of symmetry, inducing consistently small Chamfer distance between the transformed bottle ($\mathbf{T}_{1,2}\mathbf{P}_{1i}$) and the target point cloud.

where $\Psi(\cdot, \cdot)$ is Horn’s method [11] with the closed-form solution of the relative SE(3)-transform between two point clouds with known correspondence.

B. Dealing with Pose Ambiguity

SE(3)-equivariance is desirable for unraveling the relative transform between the two frames where the same object is observed. However, shape symmetry can result in ambiguity in the inferred transform, causing our latent code to be fallible when the transform selected is one of many possibilities instead of the correct one. To make our representations applicable to a broader range of objects, we therefore propose separate training objectives for object shapes with and without ambiguity w.r.t. the camera viewing frustum.

Unambiguous Objects. For objects without pose ambiguity (e.g., mugs with a handle), the transform (\mathbf{R}, \mathbf{t}) obtained from Eq. 5 should be unique and thus approximating the true inter-frame camera transform. We therefore simply minimize the L_2 distance between the estimated transform ($\hat{\mathbf{R}}_{3 \times 3}, \hat{\mathbf{t}}_3$) and the ground truth ($\mathbf{R}_{3 \times 3}, \mathbf{t}_3$):

$$L_{transform}^{uab} = \|(\hat{\mathbf{R}}\hat{\mathbf{R}}^T) - \mathbf{I}_{3 \times 3}\|_F^2 + \|\hat{\mathbf{t}} - \mathbf{t}\|_2^2, \quad (6)$$

where $\|\cdot\|_F^2$ is the Frobenius norm.

Ambiguous Objects. We limit “ambiguous objects” to objects with pose ambiguity from its shape (e.g., upright wine bottles), but not the ones that may appear ambiguous due to occlusion (e.g., mugs with their handles obscured).

Since ambiguous objects have multiple or infinite possible transforms that can meet the current observation, the exact single correct transform can never be learned. We instead wish that the derived transform will always lead to similar object shapes when transforming the object’s point cloud from one frame to another. In a nutshell, we require the latent code \mathbf{z} to implicitly learn the distribution of the possible transforms.

Hence, given the full object point clouds in two frame coordinates, \mathbf{P}_{o1} and \mathbf{P}_{o2} (readily available as we train fully in simulation), we enforce that the Chamfer distance between the two point clouds should be small after aligning them with the predicted transform:

$$L_{amb} = CD(\mathbf{T}_{1,2}\mathbf{P}_{o,1}, \mathbf{P}_{o,2})$$

$$CD(\mathbf{P}_1, \mathbf{P}_2) = \frac{1}{|\mathbf{P}_1|} \sum_{\mathbf{x} \in \mathbf{P}_1} \min_{\mathbf{y} \in \mathbf{P}_2} \|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{1}{|\mathbf{P}_2|} \sum_{\mathbf{y} \in \mathbf{P}_2} \min_{\mathbf{x} \in \mathbf{P}_1} \|\mathbf{x} - \mathbf{y}\|_2^2. \quad (7)$$

We can recover the exact transform that simultaneously justifies all current object observations by intersecting the distributions of possible transforms for multiple ambiguous objects (see Fig. 3(a) for the reasoning of the base 2-object case concerning two bottles), or further refine the predicted one when working together with unambiguous objects. Note here we do not account for the rare degenerate case of colinear axes of symmetry for all visible objects.

Furthermore, to facilitate the learning of the underlying distribution, we further augment the original ($\mathbf{P}_{10}, \mathbf{P}_{20}$) pair to include extra samples in the distribution. Given camera view \mathbf{T}_1 and \mathbf{T}_2 , we fix \mathbf{T}_2 and generate N random transforms \mathbf{T}_{1i} s that allow for camera movement around the object’s axis of symmetry with seemingly identical observations as that from \mathbf{T}_1 (Fig. 3(b)). The resulting N object point clouds in corresponding camera frames, \mathbf{P}_{1i} s, should retain similar shapes to \mathbf{P}_2 using the predicted transform. Hence, the ultimate training objective for ambiguous objects is:

$$L_{transform}^{amb} = \sum_{i=0}^N CD(\mathbf{T}_{1,2}\mathbf{P}_{1i}, \mathbf{P}_{20}), \quad (8)$$

where N and values of \mathbf{T}_i are determined by the type, e.g., cylindrical (360°) or cubical (180°), of object ambiguity. Here in our experiment, we set $N = 180$ and draw transforms from $[0^\circ, 360^\circ]$ circulation around the cylindrical bottles.

Finally, the target inter-frame transform can be similarly obtained using Eq. 5, with the two latent code \mathbf{z} s formed by concatenating all corresponding \mathbf{z}_i s of covisible objects in each frame.

C. Shape Consistency across Viewing Angles

Since \mathbf{z}_0 is SO(3)-equivariant, its rotation invariant part, $\mathbf{s} \in \mathbb{R}^k$, encoding full object shapes, can then be extracted as $\mathbf{s} = \{s_i\}_{i=1}^{i=k} = \|\mathbf{z}_0\|_2$, where we term \mathbf{s} as the shape descriptor.

Following [10], we adopt the batch-hard shape similarity loss L_{b_shape} , enforcing \mathbf{s} to be consistently similar across viewing angles of the same object while discriminatively far apart for different objects.

L_{b_shape} takes the form of the triplet loss as [anchor, positives, negatives]. To allow for a variety of viewing angle combinations during training, we populate each training batch B with M partial observations for each of the N randomly drawn objects. Samples of the same object instance serve as

mutual *anchors* and *positives*, $(\mathbf{A}_i, \mathbf{P}_i)$, with samples not from the current shape instance being the *negatives*, \mathbf{N}_i . L_{b_shape} is calculated in a “batch-hard” fashion, i.e., it only uses the most dissimilar (\mathbf{A}, \mathbf{P}) and the most similar (\mathbf{A}, \mathbf{N}) for each anchor to guide the training. With $D(\cdot, \cdot)$ as the cosine similarity, the final batch-hard shape similarity loss is formulated as:

$$L_{b_shape} = \frac{1}{|B|} \sum_{i=1}^N \sum_{j=1}^M \left(- \min_{k \in [1, M]} D(o_{ij}, o_{ik}) \right. \\ \left. + \max_{m \neq i} D(o_{ij}, o_{mn}) \right),$$

where o_{ij} is the j th observation of object i within the batch.

D. Training in Simulation

Training Objective. NeuSE is trained with partial object point clouds and corresponding 3D occupancy voxel grids of objects’ complete geometry. The full model $[f_\theta, \Phi]$ predicts the complete 3D occupancy values at query object locations, which is then evaluated by the standard cross-entropy classification loss $L_{occ} = \mathcal{L}(\Phi(\mathbf{p}, f_\theta(\mathbf{P}), v))$ with sampled query location \mathbf{p} and its corresponding true occupancy value v .

The ambiguous and unambiguous object categories are trained separately, with respective $L_{transform}$ and shared L_{occ} and L_{shape} . The final training objective is the weighted sum of the three losses

$$L = L_{occ} + \beta_1 L_{transform} + \beta_2 L_{b_shape}, \quad (9)$$

where β_1 and β_2 are constants set to balance the order of magnitude of the three losses. The training samples are organized following L_{b_shape} ’s formulation, where L_{occ} is evaluated for each sample in B and $L_{transform}$ for any two observations of the same object. With this composition of the training data, the model is expected to see various pairs of viewing angles and learn to predict the relative transform between two frames within a certain range apart.

Data Generation. NeuSE is trained fully in simulation with RGB-D images rendered with Pybullet [6]. We place a randomly-posed principal object on the table, along with 2-4 (for unambiguous objects) and 1-2 (for ambiguous objects) objects arbitrarily selected from the trained categories to simulate a typical cluttered environment. In light of the viewing angle variety, for each multi-object layout, we uniformly sample a fixed number of camera locations over the hollow cubical space centered around the table. The cubical space is set to be $[d_n, d_f]$ away from the table within the table plane and $[d_l, d_h]$ away from the table in the vertical direction, thus accounting for observations from near, far, low, and high locations.

IV. NEUSE-BASED OBJECT SLAM WITH LONG-TERM SCENE INCONSISTENCIES

NeuSE enables robust data association across viewing angles and further serves as a lightweight, alternative “sensor” for providing cross-frame camera pose constraints. We propose a NeuSE-based localization strategy in tandem with a change-aware object-centric mapping procedure to enable robust robotic operation in scenes with long-term changes.

A. System Formulation and Update

Our object-based SLAM problem is formulated as a pose graph consisting of only keyframe camera pose vertices, where an edge exists to constrain the two vertices if there are inter-frame transform measurements available from NeuSE or any other sources. The measurement error between vertex i and j for each edge is defined as $\mathbf{e}_{ij} = \log(\mathbf{Z}_{ij} \hat{\mathbf{T}}_j^{-1} \hat{\mathbf{T}}_i)^\vee$, where \mathbf{Z}_{ij} is the odometry measurement from arbitrary sources between frame i and j , and $\hat{\mathbf{T}}$ is the current estimate of \mathbf{T} .

The system maintains a library of keyframes with the latest camera pose estimates obtained via pose graph updates, as well as NeuSE latent codes of the observed objects in the frame coordinate. The camera pose of the current frame is recovered as the smoothed estimate of pose constraints from associated objects and external sources between the frame itself and the nearest keyframe.

The objects in the system are recorded by their per-keyframe visibility, change status, a partial point cloud from their last keyframe observation (for query points generation during rendering), and the latest shape descriptor from initialization or mapping updates.

For localization, the system works only with latent codes in the local camera frame, while their world-frame counterparts are used for mapping operations. When an object is first observed, its world-frame latent code is initialized and then updated as needed by averaging the back-projected latent codes of the same object using the latest camera pose estimates recorded in the keyframe library.

B. Data Association

NeuSE-predicted inter-frame transforms are only valid if computed from latent codes belonging to the same object. Our data association scheme exploits both full shape similarity and spatial proximity so as to allow pose constraint generation only between latent codes with reliable object association.

Shape Similarity. For each object in the current frame, we extract the shape descriptor from the latent code and calculate its cosine shape similarity (as adopted in Eq. III-C) with all objects in the library. Objects with a shape similarity score greater than δ_{shape} are considered potential data association candidates \mathcal{O}_c . If no similarity scores exceed δ_{shape} , a new object instance is initialized and added to the object library.

Spatial Proximity. Spatial proximity involves examining the Euclidean distance between the partial point cloud center of the current object and its candidates in \mathcal{O}_c , where the current partial center is projected to the latest keyframe its candidate is last seen. The transform for projection is calculated using Horn’s method (Eq. 5) between corresponding latent codes. Candidate with the smallest distance while below δ_{prox} is deemed a successful match to be included in $\mathcal{O}_{matched}$ for further pose constraint generation. Otherwise, the current object is unassociated and grouped into $\mathcal{O}_{unmatched}$.

The procedure is performed first on unambiguous objects and later on ambiguous objects, differing only in the acquisition of inter-frame transforms. For unambiguous objects, we compute the transform directly using Horn’s method. For

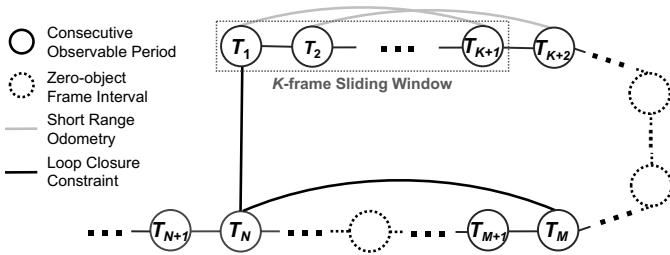


Fig. 4: **Pose graph optimization.** With objects observed in periods of consecutive frames, we derive from corresponding latent codes (1) short range odometry constraints (grey) within a local K -frame sliding window, and (2) global loop closure constraints (black) between the current (\mathbf{T}_N) and the first frame of each of its previous consecutive observable periods (\mathbf{T}_1 and \mathbf{T}_M), which are working jointly to constrain the pose graph optimization.

ambiguous objects, we utilize the transform from associated unambiguous objects if available. If not, we conduct an exhaustive search of all paired combinations of covisible object candidates in previous keyframes and obtain the inter-frame transform from the concatenated object latent codes.

We hence divide all covisible objects \mathcal{O} in one frame into three groups: (1) $\mathcal{O}_{matched}$, which has objects with shape and spatial consistency, and is adopted for pose constraint generation, (2) $\mathcal{O}_{unmatched}$, which consists of scene changes or temporally ambiguous observations, and is processed by change detection, and (3) new objects never seen before.

C. Pose Graph Optimization

With objects successfully associated across frames, we compute NeuSE-predicted transforms among frames so as to constrain the pose graph both locally and globally (see Fig. 4).

Keyframe Selection. Keyframes are selected based on the presence of new objects and proximity to previous keyframes. New objects trigger the selection of a frame as a keyframe, and frames located at least 0.04m away from the previous keyframe based on accumulated odometry are also chosen. Additional keyframes may be added after change detection for frames with changes.

Short-range Odometry. To reduce local drift in frames with persistently observed objects, short-range NeuSE-predicted pose constraints are applied to a sliding window optimization of K keyframes. For each newly added keyframe, we search its preceding $K - 1$ keyframes and identify the common objects observed between the current and previous frames. The inter-frame transform constraint is computed based on the concatenated latent codes of the shared objects (or a single unambiguous object) and then added as an edge to the pose graph.

Long-range Loop Closing. Global loop closing is activated when an object is detected again in a frame after its last consecutive observable period. The common objects between the current frame and the initial frames of all its previous observation periods are identified, and relative transform constraints are derived from the concatenated NeuSE latent codes.

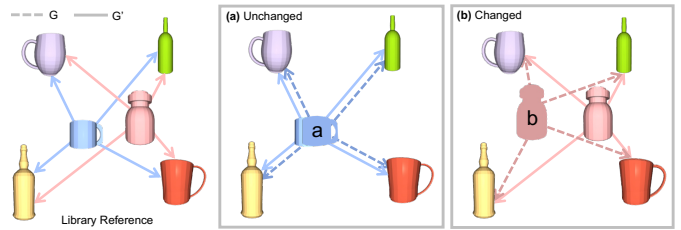


Fig. 5: **Object layout comparison through graph matching.** Object graphs are constructed for the current frame (G) and the library (G'). For object a and b , which are similar in shape to the blue mug and pink bottle in the library, respectively, the inter-object distance between them and the anchor objects in the four corners are computed and compared. (a) All corresponding edges (dashed and solid lines) with anchor objects have similar oriented lengths, indicating that the mug is unchanged but was seen with an occluded handle, leading to a false ambiguous transform by the latent code. (b) There are no similar edges, indicating a different layout with the bottle moved.

These constraints are then added to the pose graph, which initiates a global optimization process using the latest pose estimates from the local sliding-window optimization as the starting point.

D. Change-aware Object-centric Mapping

Change detection is performed frame-by-frame on objects in $\mathcal{O}_{unmatched}$ that match in shape but are identified as spatially apart based on latent codes, providing a foundation for consistent long-term mapping.

As changes are often gradual and occupy a small portion of the object clutter in long-term scenes, here change detection is done by comparing the relative layout of the query unmatched object $o_{ui} \in \mathcal{O}_{unmatched}$ with all objects $o_{mi} \in \mathcal{O}_{matched}$ in the matched set serving as anchors. We argue that the relative object position disparity is more robust to camera pose drift compared to the absolute position difference, as all objects observed will be drifting concurrently in the world frame.

We represent the local layout with a directed object graph G constructed with $\mathcal{O}_{unmatched}$ and $\mathcal{O}_{matched}$. Each vertex of G represents an object o with its shape descriptor and the true object center as (s, c) . The center c is computed from the full object reconstruction using the decoding steps in Eq. 4 and back-projected to the world frame using the latest camera pose estimate. Edges are established between objects $o_{ui} \in \mathcal{O}_{unmatched}$ and all anchor objects $o_{mj} \in \mathcal{O}_{matched}$, indicating the oriented distance between their centers $E = \{e_{ij} | e_{ij} = c_{ui} - c_{mj}, \forall o_{ui} \in \mathcal{O}_{unmatched}, o_{mj} \in \mathcal{O}_{matched}\}$.

We build the local and reference object graph, G and G' , respectively, for \mathcal{O} in the current frame and their associated or shape-similar counterparts in the system library (see Fig. 5). After a quick alignment of the two graphs using the centers of anchor objects, for each pair of edges $(e_{ij}, e_{i'j'})$ connecting vertices of similar shapes (determined by s from data association), we compare their edge value disparity to assess if this

is a changed layout:

$$\sum_j \mathbb{1}(|\mathbf{e}_{ij} - \mathbf{e}_{i'j'}| \leq \delta_e) = \begin{cases} 0, & \text{changed} \\ \geq 1, & \text{unchanged.} \end{cases} \quad (10)$$

An object o_i is marked as *unchanged* if at least one pair of edges is found to be closer than a threshold δ_e . This indicates that its inter-spatial relationship with at least one of the anchor objects is consistent. If no edges are found to be close, the object is marked as *changed* and its change status and partial point cloud are updated in the object library. Here, we define an object to be “removed” from the scene if it has never been shape-matched in frame periods with global loop closure.

Therefore, we are able to maintain a lightweight, object-centric map that accurately reflects the full object reconstructions from NeuSE predictions. By using objects as the basic building blocks of the map, we can update changes seamlessly by replacing the old latent code with the new one during the decoding stage, avoiding the cumbersome and artifact-prone point- or voxel-wise modifications commonly used in traditional low-level geometric maps.

V. EXPERIMENTS AND RESULTS

We aim to assess the efficacy of NeuSE for object shape and pose characterization and robot spatial understanding. Specifically, we would like to answer two questions: **(1)** Can NeuSE-based object SLAM perform reliable localization on its own or improve existing results when combined with other SLAM measurements, especially in the presence of temporal scene inconsistencies? **(2)** Can the proposed approach build consistent object-centric environment map with timely updates to reflect scene changes? We train NeuSE fully in simulation, and evaluate the proposed algorithm directly on both synthetic and real-world sequences consisting of unseen objects of the trained categories, where objects are added, removed, and switched places to simulate long-term environment changes.

A. Datasets

Given the limited availability of object model collections for training and the scarcity of public data with appropriate object-level scene changes, we created our own synthetic and real-world sequences. The collected data feature mugs and bottles in various cluttered arrangements, with diverse occlusion patterns, various viewing angles, and gradual object changes. We chose mugs and bottles as the representative object categories due to their common use and distinct unique (mugs) or ambiguous cylindrical (bottles) shapes for localization, which allow us to evaluate the effectiveness of our latent code design. Following past work [35, 23, 28], our approach should be extendable to even more categories by incorporating related objects into training.

Synthetic Sequences. An environment is rendered in Pybullet with 50 previously unseen ShapeNet [3] mugs and bottles scattered onto ten tables in a 10×15 (m) area (Fig. 6(a)). To fully examine the proposed SE(3)-equivariance of NeuSE, two object layouts are generated: (1) a roughly planar layout with all upright objects, and (2) a non-planar hilly layout

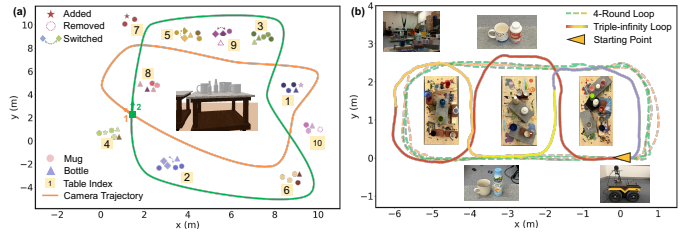


Fig. 6: **Evaluation data overview.** Object changes happen at each joint of the colored trajectory segments. **(a)** Table layout with object changes and the ground truth camera trajectory of the two synthetic sequences. **(b)** Real-world setup with ground truth camera trajectories.

with nearly half of the objects laid down and arbitrarily oriented on tabletops. The camera follows a preset closed-loop trajectory and records RGB-D images and segmentation masks of both layouts, respectively. This leads to two sequences with *uninterrupted* object observation among the ten tables, where objects are revisited on most tables (excluding table 4, 7, and 10) from approximately opposite views. For each sequence, objects are added, removed, or moved to different locations, resulting in a total of nine changes within the trajectory.

Real-world Sequences. 28 mugs and bottles of various shapes and sizes are densely located on five tables in a 6×3 (m) space (Fig. 6(b)), among which ten objects are added, removed, or switched locations to create two sets of object arrangements. A RealSense D515 camera mounted on a Clearpath Jackal robot records RGB-D data along two preset trajectories: (1) A four-round peripheral loop around three central tables, with the first two rounds captured with one object arrangement and the latter two with the other arrangement, in total having nine changed objects. (2) A more challenging triple-infinity loop where the camera moves through four central and side tables, with seven object changes along the way. The ground truth camera trajectories are recovered from a Vicon motion capture system. The object segmentation masks are obtained from Detectron2 [42].

B. Implementation Details

To train NeuSE’s occupancy network, we generate two sets of training samples using 94 mug models and 242 cylindrical bottle models from ShapeNet. The sets are respectively for unambiguous (mugs) and ambiguous (bottles) objects, each containing 60,000 RGB-D partial observations with segmentation masks. We follow the sample generation strategy in Section. III-D: 2000 object layout mixing bottles and mugs are created in Pybullet, from each of which 30 views are uniformly sampled with $[d_n, d_f] = [0.3, 5]$ (m) and $[d_l, d_h] = [-0.2, 0.2]$ (m). We train our approach on two NVIDIA RTX 3090 GPUs using a learning rate of 5×10^{-4} with the Adam optimizer. The latent code size is $k = 512$ and the occupancy threshold for reconstruction is $v_0 = 0.5$. We set the weight coefficients in Eq. 9 to be $(\beta_1, \beta_2) = (0.1, 0.1)$ for unambiguous objects, and $(\beta_1, \beta_2) = (1, 0.1)$ for ambiguous objects. The training batch is populated with eight object shapes, each with 15 partial observations, by setting $M = 15$ and $N = 8$.

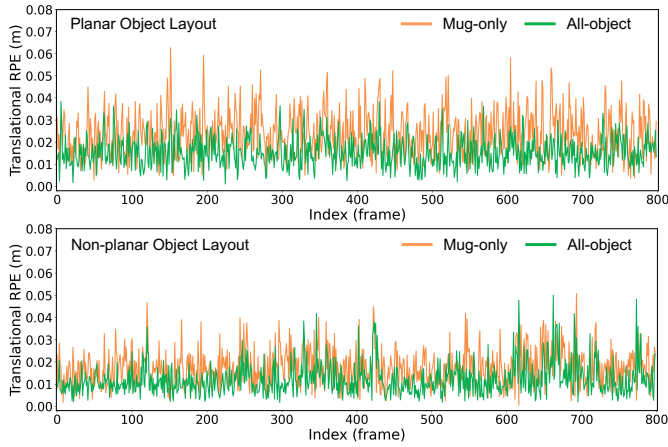


Fig. 7: **Distribution of translational RPE along synthetic sequences.** The green lines in both layouts reveal lower RPE dispersion, indicating consistently lower local drift when using all objects of interest (mugs and bottles), as opposed to using only unambiguous ones (mugs).

For the object SLAM system, we have $\delta_{shape} = 0.95$, $(\delta_{prox}, \delta_e) = (0.03, 0.02)$ (m) for the synthetic sequence, and $(\delta_{prox}, \delta_e) = (0.04, 0.03)$ (m) for real-world sequences for data association and change detection. We set the sliding window size as $K = 10$ and adopt the factor graph representation for SLAM pose graph optimization. The local sliding-window optimization is solved with a Levenberg–Marquardt fixed-lag smoother, and the global pose graph is solved with iSAM2 [15], both using implementations from GTSAM [7].

C. Localization with Temporal Scene Inconsistencies

All results are obtained on a laptop with an Intel Core i7-9750H CPU and an Nvidia GeForce RTX 2070 GPU. NeuSE network inference takes 6ms per object, with inter-frame pose constraint calculation taking 1ms. One-time rendering for object-centric map construction costs 30ms per object with 20,000 query points. With data association included, the speed is approximately 28fps for generating object-level inter-frame pose constraints with our NeuSE-based front-end, making it promising for NeuSE to be integrated as an external “constraint sensor” with real-time operating speed. The final overall localization speed of our change-aware SLAM system is 11fps for the current experiment setting, with no software optimization or major tuning of the back-end iSAM2 solver. All following localization results are reported as the median of five runs.

Synthetic Sequences. The consecutive observations of objects in the synthetic data allow for uninterrupted operation of the proposed SLAM strategy, enabling an independent evaluation of NeuSE’s capabilities for conducting change-aware localization and mapping.

Therefore, we report quantitatively in Table. I the Root Mean Squared Error (RMSE) of both the translational Relative Pose Error (RPE) and the Absolute Trajectory Error (ATE) of the estimated camera poses for the two testing sequences, showcasing consistent NeuSE’s performance both locally and

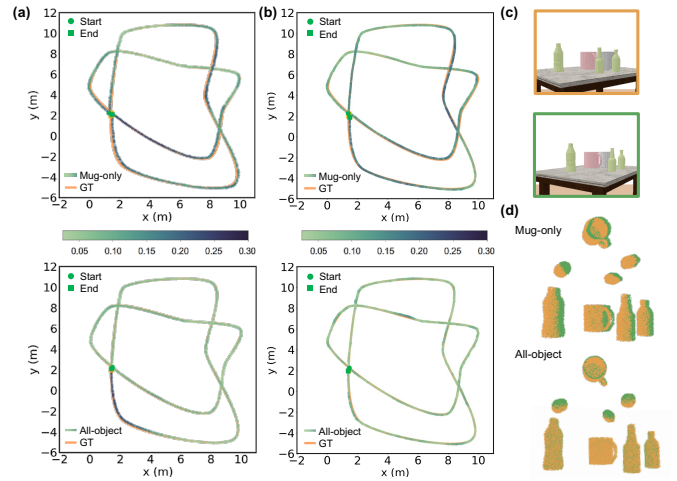


Fig. 8: **Column 1-2: Comparison of estimated and ground truth trajectories (GT) on synthetic sequences.** (a) Planar and (b) Non-planar object layout. Color variation implies ATE value distribution along the path. All-object leads to better estimation accuracy than Mug-only, as shown by the evenly lighter trajectory color with lower ATE values. **Column 3: Ambiguous objects for inter-frame transform prediction.** With object point clouds in (c) transformed from the orange frame to the green frame using transforms derived from merely pink mugs and together with green bottles, the better point cloud alignment in (d) of All-object over Mug-only demonstrates the effectiveness of using covisible ambiguous objects to improve transform prediction accuracy.

globally. We further visualize the RPE and ATE error distribution along the way in Fig. 7 and Fig. 8, respectively.

To justify our treatment of the inclusion of ambiguous objects, we run two variants of the system as (1) Localizing with mugs only (Mug-only), and (2) Localizing with all objects of interest, i.e., mugs and bottles (All-object). For the few frames with no objects for data association or pose generation, we maintain system operation with odometry measurements corrupted from ground truth by a zero-mean Gaussian noise with $\sigma = 0.003$ (rad) for rotation and $\sigma = 0.05$ (m) for translation.

The RPE and ATE values in Table. I show that (1) NeuSE is a reliable “constraint sensor” for producing consistent short- and long-range camera pose constraints, and (2) our system is capable of producing a globally consistent trajectory, despite various occlusion patterns, viewing angle disparities, and object changes between the two traversals. The smooth distribution of RPE throughout the sequence, as shown in Fig. 7, also demonstrates the robustness of our localization strategy against temporal scene changes, which is attributed to the effectiveness of our proposed data association and change detection in distinguishing objects in the second traversal.

Specifically, we observe from Table I that the proposed object SLAM approach performs better on the non-planar object layout, fully showing the efficacy of our SE(3)-equivariant representations in handling randomly oriented objects. This can be attributed to our training data generation strategy, which

TABLE I: RMSE of ATE and Translational RPE on synthetic sequences. Gains (Δ) are computed based on results from Mug-only. Best results are marked in bold.

	Planar			Non-planar		
	Mug-only	All-object	Δ (%)	Mug-only	All-object	Δ (%)
RMSE of ATE (m)						
Synthetic: 1 st traversal	0.072	0.043	40.3%	0.058	0.045	22.4%
Synthetic: 2 nd traversal	0.096	0.071	26.0%	0.077	0.033	57.1%
Synthetic: Full	0.116	0.065	44.0%	0.091	0.053	41.8%
RMSE of Trans RPE (m/f)						
Synthetic: Full	0.026	0.017	34.6%	0.024	0.016	33.3%

includes various views and occlusion patterns to learn robust geometric features of object shapes across viewing angles. Further, the lying-down mugs in the sequence help reduce shape ambiguity by providing more valid observations for generating camera pose constraints, as their handles are more frequently visible when pointing upwards than in the usual sideways direction. With the SE(3)-equivariant property of NeuSE, our approach can learn from upright observations to benefit the processing of laid down objects, thus enabling generalization to new scenarios with various object orientations.

Our attempt for the incorporation of ambiguous objects for pose constraint generation is validated by (1) the consistent improvement of All-object over Mug-only throughout the two traversals in Table. I, and (2) the lower dispersion of RPE values for All-object in Fig. 7. Besides, in Fig. 8(c)-(d), with object point clouds in (c) transformed from the upper (orange) to the lower (green) frame using transforms derived from only the pink mug and together with green bottles, the better point cloud alignment in (d) of All-object over Mug-only demonstrates the viability of leveraging covisible ambiguous objects for improving transform estimation accuracy.

Real-world Sequences. It is common for objects to be out of sight during real-world robot motion. Hence, in this section, we validate the feasibility and benefit of our strategy in complementing other SLAM measurements and promoting loop closing for a globally consistent estimated trajectory.

In this spirit, we adopt ATE as the metric and compare our approach to two directly deployable object-based SLAM strategies, CubeSLAM [44] and EM-Fusion [37], as well as the popular and state-of-the-art ORB-SLAM3 [2] pipeline. CubeSLAM assumes a static operating environment (or objects with known motion models, which is not applicable here) and EM-Fusion can handle moving objects in the scene. They serve as baselines to evaluate object SLAM performance and the potential influence of object changes in the scene. For CubeSLAM, the implementation of its integration with ORB-SLAM3 is chosen. As ORB-SLAM3 does not address temporal scene inconsistencies, to explore the effect of object changes onto localization performance, we generate two sets of ORB-SLAM3 odometry measurements as baselines by running it (1) non-stop (ORB3-NS) for the whole trajectory, and (2) piecewise (ORB3-PW) for each trajectory segment with consistent object layout (as shown in Fig. 6(b)).

In addition, to verify NeuSE’s transferability from sim-

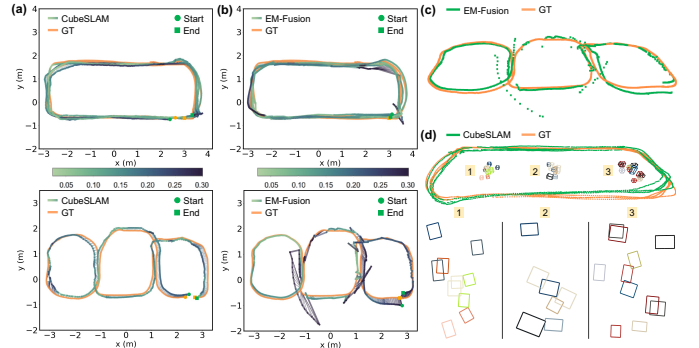


Fig. 9: **Column 1-2: Visualization of the estimated trajectories: (a) CubeSLAM and (b) EM-Fusion.** Color variation of the line indicates ATE value distribution along the trajectory. **Column 3: Trajectory/object cuboid estimation drifts of the two selected object SLAM baselines. (c):** EM-Fusion undergoes heavy out-of-plane drift in the Triple-infinity loop due to faster rotations around the corners. **(d):** The top-down view (bottom row) displays the cuboid estimates of mugs and bottles in the 4-Round loop. CubeSLAM struggles to handle object changes, which causes inaccuracies in data association. This results in multiple missed, drifted, and falsely overlapped cuboid detections and affects the joint optimization of cuboid estimates and camera trajectory.

ulation to reality, we follow the object-only experiments for synthetic data and run Mug-only and All-object on the two real-world sequences. Raw Odometry, generated using Open3D [47] based on photometric and geometric loss [27], are adopted to sustain system operation when no objects are in sight or associated to generate a pose constraint.

We present in Table. II the RMSE of ATE for all estimated trajectories and visualize them in Fig. 9 and Fig. 10. The tracking failure of the 4-Round loop (the two spikes in the first row of Fig. 10) are excluded from RMSE calculation to better reflect the global localization performance of the trajectory.

The transferability of NeuSE from simulation to the real world is verified by its fair performance in terms of RMSE values and remarkable correction of the accumulated drift from Raw Odometry, as seen in the first column of Fig. 10. This confirms NeuSE’s full functionality when applied to real data.

In comparison to the two selected object SLAM baselines that use all detected objects in the scene, from Table. II and Fig. 9, our proposed approach outperforms CubeSLAM and

TABLE II: RMSE (m) of the Absolute Trajectory Error on real-world sequences. Best results for each trajectory are marked in bold.

	CubeSLAM [44]	EM-Fusion [37]	Obj-only			ORB3-NS		ORB3-PW	
	All Objects Detected	All Objects Detected	Raw Odometry	Mug-only	All-object	Base	+ Ours	Base	+ Ours
4-Round: 1 st – 2 nd round	0.108	0.162	1.22	0.122	0.112	0.101	0.096	0.102	0.084
4-Round: 2 nd – 3 rd round	0.114	0.174	1.85	0.124	0.114	0.126	0.090	0.102	0.083
4-Round: 3 rd – 4 th round	0.128	0.127	2.07	0.123	0.090	0.119	0.085	0.086	0.076
4-Round: Full	0.131	0.154	3.51	0.134	0.111	0.118	0.092	0.093	0.079
Triple-infinity	0.147	0.193	1.12	0.137	0.106	0.101	0.082	0.160	0.083

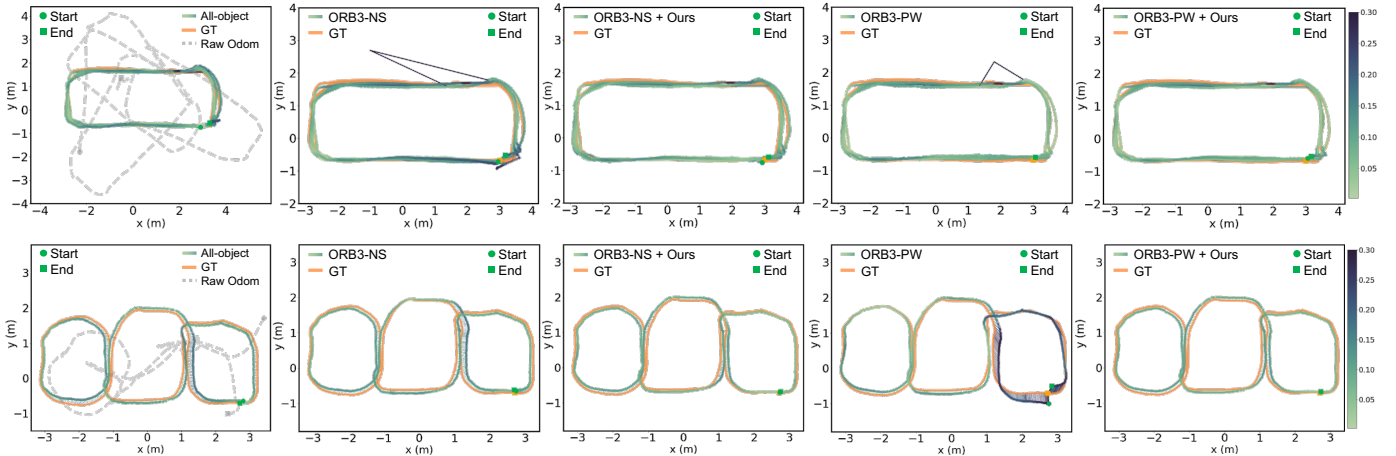


Fig. 10: **Visualization of estimated trajectories against ground truth (GT).** Color variation (color bar on the right) of the line indicates ATE value distribution along the trajectory. Above: Estimated trajectories of the 4-Round loop. The integration of our strategy (column 3 and 5) helps prevent the tracking failure, as shown by the two spikes in the second and fourth column. Below: Estimated trajectories of the Triple-infinity loop. Our strategy (column 5) successfully eliminates the start and end point drift for ORB3-PW (column 4), resulting in improved trajectory estimate when revisiting the rightmost table, as indicated by the lighter color of ATE values along the trajectory.

EM-Fusion on both the 4-Round and Triple-infinity loop with using all objects of our interest (mugs + bottles), showing the advantage of NeuSE for facilitating lightweight and robust localization in real-world sequences with scene inconsistency.

Notably, CubeSLAM produces shrinking camera trajectory estimate in Fig. 9(a) around the right table with object changes in the Triple-infinity loop, and multiple missed, drifted, and falsely overlapped cuboid estimates from the top-down view (bottom row) in Fig. 9(d). Assuming a static environment, CubeSLAM struggles to address object changes within the two sequences, inducing errors in cuboid association and estimation among old and new objects in neighboring areas. This yields false camera-cuboid geometric constraints, and ultimately affects the jointly optimization of object cuboids and camera trajectory.

Meanwhile, EM-Fusion, as shown in Fig. 9(b) and (c), gives subpar bumpy and drifted trajectory estimates. While it can handle scene layout changes at sequence segment intersections, EM-Fusion suffers from lower tracking accuracy due to accumulated drift from less object overlap. Besides, originally tested on tabletop scenes, EM-Fusion requires a coarser SDF background volume resolution so as to avoid memory exhaustion here in our larger multi-table scenario, leading to a further loss of accuracy in camera tracking.

As to working jointly with other SLAM measurements,

in Table. II, we observe consistent improvement in terms of RMSE values when integrating our proposed strategy (using all objects) with the vanilla ORB-SLAM3 measurements. NeuSE enables robust data association and manages to prevent the occurrence of tracking failure (the spikes in the second and fourth column of Fig. 10) for the 4-Round trajectory.

The greatest RMSE improvement in Table. II is observed from ORB3-PW + Ours on the Triple-infinity trajectory. Our proposed strategy helps decrease the RMSE by 48.1% from 0.16m to 0.083m. In this way, ORB3-PW + Ours outperforms ORB3-NS (0.101m) despite receiving less global loop closing constraints from ORB3-PW than ORB3-NS, while aligning the start and end point with better trajectory accuracy when revisiting the rightmost table. Considering the little scene overlap within each of the four trajectory segments, this notable improvement highlights the critical role of our strategy in constraining pose estimates in short and longer range, especially when insufficient loop closing (e.g., throughout ORB3-PW) is performed by the external SLAM system.

Our strategy also demonstrates robustness in handling scene changes, despite the less significant improvement in the 4-Round loop that is with abundant loop closure from ORB-SLAM3. The fourth column of Table. II presents the RMSE values of ORB3-NS on different parts of the 4-Round loop, as the sequence proceeds with object layout transition. Note

TABLE III: Change detection results on the synthetic and real-world sequences. Best results are marked in bold.

	TP	FP	FN	Pr	Re
Synthetic					
PMT	7	2	2	77.8%	77.8%
Ours	9	0	0	100.0%	100.0%
4-Round					
PMT	7	0	2	100.0%	77.8%
Ours	9	0	0	100.0%	100.0%
Triple-Infinity					
PMT	5	2	2	71.4%	71.4%
Ours	7	1	0	87.5%	100.0%

ORB3-PW does not run between the second and third round, with the corresponding value listed only for comparison purposes. When object changes happen at the intersection of the second and third round, ORB3-NS is clearly affected and shows an RMSE jump from 0.101m to 0.126m. On the contrary, our effective data association based on full object shape similarity and spatial proximity allows ORB3-NS + Ours to maintain a steady yet gradually improving RMSE (around 0.09m) during object changes, bringing ORB3-NS almost on-par performance with ORB3-PW (free from object changes) for the entire trajectory.

D. Change-aware Object-centric Mapping

Built on top of the decoding steps in Eq. 4 for full object reconstruction, we demonstrate the ability to maintain a consistent map of objects of interest in the environment, with always timely update of the latest changes.

Since there are no suitable SLAM pipelines for direct comparison of mapping with temporal scene changes, we use the recent object-level mapping method with online change detection, panoptic multi-TSDFs (PMT) by Schmid et al. [33], as our baseline. We feed PMT with our trajectory estimates that have the lowest RMSE of ATE values and compare the change detection results for synthetic and real-world sequences.

We quantify the performance of our system and PMT in Table. III by comparing the number of correctly detected changes (true positives, TP), falsely detected changes (false positives, FP), and undetected changes (false negatives, FN). We further calculate precision (Pr) and recall (Re) rates based on these numbers. The results show that our system correctly detects most of the changes for both synthetic and real-world data, while PMT produces several false positives and false negatives due to localization errors and inability to reason holistically from partial observations.

Qualitatively, we present in Fig. 11 reconstructions of all objects that have appeared in the synthetic planar and non-planar layouts, respectively. Fig. 12 displays the map evolution of our method and PMT before and after changes for each table in the real-world sequences. Our approach generates a lightweight, object-centric map that precisely captures changes (see Fig. 12(b) and (c)). In contrast, PMT, being a traditional TSDF-based mapping technique, fails to deliver

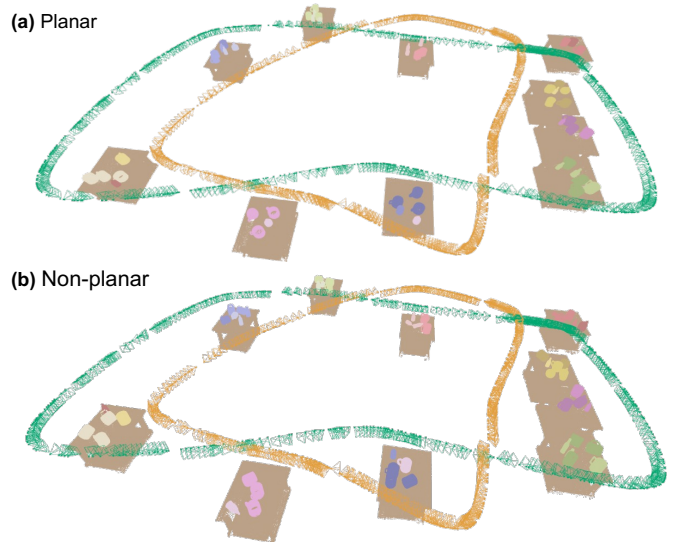


Fig. 11: **Complete object reconstruction of synthetic sequences for the two object layouts.** (a): Planar layout and (b): Non-planar layout. Tables are rendered for visual clarity, whose points are back-projected to the world using camera pose estimates from NeuSE-predicted constraints, demonstrating the effectiveness of our localization strategy.

accurate change detection results and produces reconstructions with various defects. PMT struggles to distinguish between switched objects of the same category due to its inability to perform full object shape comparison as NeuSE does. This is shown by the overlapping reconstructions of the white and green bottles (object 2 and 3 of table 3) and the red and black mugs (object 8 and 9 of table 5) in Fig. 12(d). In addition, Fig. 12(e) highlights PMT’s susceptibility to localization errors, where it mistakenly marks the green mug on Table 5 as newly added when the other side of the mug, which is observed later, drifts to be misaligned with the original volume.

VI. CONCLUSION

In this paper, we present NeuSE, a category-level neural SE(3)-equivariant embedding for objects, and demonstrate how it supports object SLAM for consistent spatial understanding with long-term scene inconsistencies. NeuSE differs itself from prior neural representations adopted in SLAM through its ability to *directly* obtain camera pose constraints from SE(3)-equivariance and its flexible map representation that easily accommodates long-term scene changes. Our evaluation results on both synthetic and real-world data showcase the feasibility of our approach for change-aware localization and mapping when working stand-alone or as a complement to traditional SLAM pipelines.

ACKNOWLEDGMENTS

The authors thank Shichao Yang for discussion and guidance in setting up CubeSLAM for the real-world testing sequences. This work was supported by ONR MURI grant N00014-19-1-2571 and ONR grant N00014-18-1-2832.

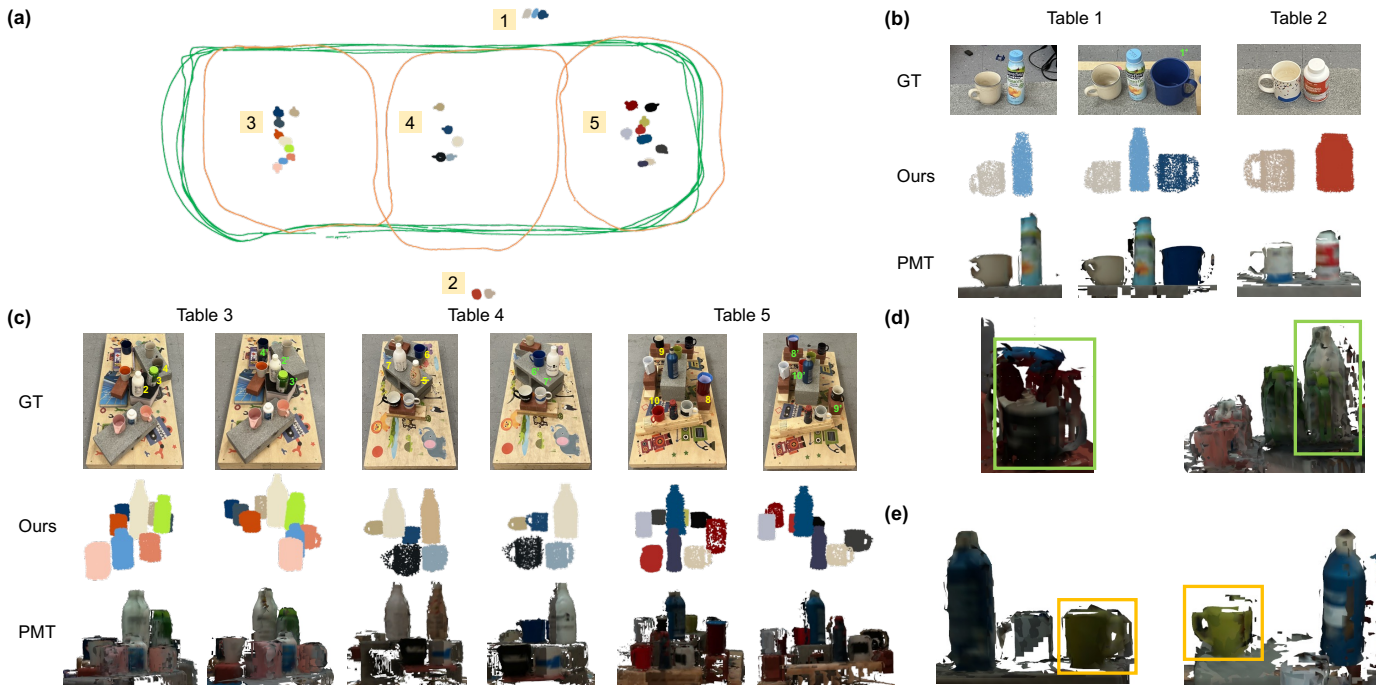


Fig. 12: **Results of change-aware mapping for the real-world sequences.** (a) Comparison of our object-centric map to ground truth trajectories, displaying qualitative spatial consistency. (b) and (c) show the evolution of the reconstructed object layout before and after changes, with ground truth scenes (GT), object-centric maps from our approach (Ours), and PMT reconstructions (PMT) from top to bottom. Changed objects are numbered as n , with n' representing their correspondence after changes or newly added objects, and $\#$ indicating objects removed from the scene. (d) Reconstruction artifacts of overlapping bottles (left) and mugs (right) from PMT's change detection failure. (e) False positive changed mug marked by PMT due to imperfect localization, where little overlap exists between the two sides of the green mug when viewed from different frames.

REFERENCES

- [1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. In *RA-L*, 2022. 2
- [2] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1, 9
- [3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 7
- [4] Ethan Chun, Yilun Du, Anthony Simeonov, Tomas Lozano-Perez, and Leslie Kaelbling. Local neural descriptor fields: Locally conditioned object representations for manipulation. *arXiv preprint arXiv:2302.03573*, 2023. 2
- [5] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbee-SLAM: A real-time monocular visual SLAM with ORB features and NeRF-realized mapping. *arXiv preprint arXiv:2209.13274*, 2022. 2
- [6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016. 5
- [7] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. 8
- [8] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12200–12209, 2021. 2, 3, 15
- [9] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part II 13*, pages 834–849. Springer, 2014. 1
- [10] Jiahui Fu, Yilun Du, Kurran Singh, Joshua B. Tenenbaum, and John J. Leonard. Robust change detection based on neural descriptor fields. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2817–2824, 2022. 2, 4
- [11] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642, 1987. 3, 4

- [12] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-time monocular object-model aware sparse SLAM. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7123–7129. IEEE, 2019. 1, 2
- [13] Jeffrey Ichnowski*, Yahav Avigal*, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *CoRL*, 2020. 2
- [14] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. In *RSS*, 2021. 2
- [15] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, 2011. doi: 10.1109/ICRA.2011.5979641. 8
- [16] Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th Annual Conference on Robot Learning*. 2
- [17] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86, 2009. doi: 10.1109/ISMAR.2009.5336495. 1
- [18] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 2
- [19] Cheng-Wei Lin, Tung-I Chen, Hsin-Ying Lee, Wen-Chin Chen, and Winston H Hsu. Coarse-to-fine point cloud registration with se (3)-equivariant representations. *arXiv preprint arXiv:2210.02045*, 2022. 2
- [20] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In *6th Annual Conference on Robot Learning*, 2022. 2
- [21] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017. 1, 2
- [22] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level SLAM. In *2018 international conference on 3D vision (3DV)*, pages 32–41. IEEE, 2018. 1, 2
- [23] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. CVPR*, 2019. 2, 7
- [24] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanculescu, and Arnaud de La Fortelle. Lens: Localization enhanced by nerf synthesis. In *Conference on Robot Learning*, 2022. 2
- [25] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018. 1, 2
- [26] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *RSS*, 2022. 2
- [27] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 143–152, 2017. doi: 10.1109/ICCV.2017.25. 9
- [28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, 2019. 2, 7
- [29] Antoni Rosinol, John J Leonard, and Luca Carlone. NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. 2, 3
- [30] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018. 1, 2
- [31] Hyunwoo Ryu, Jeong-Hoon Lee, Hong-in Lee, and Jongeun Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. *arXiv preprint arXiv:2206.08321*, 2022. 2
- [32] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013. 1, 2
- [33] Lukas Schmid, Jeffrey Delmerico, Johannes Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 11
- [34] Bokui Shen, Zhenyu Jiang, Christopher Choy, Leonidas J Guibas, Silvio Savarese, Anima Anandkumar, and Yuke Zhu. Acid: Action-conditional implicit visual dynamics for deformable object manipulation. *arXiv preprint arXiv:2203.06856*, 2022. 2
- [35] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipula-

- tion. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400, 2022. doi: 10.1109/ICRA46639.2022.9812146. [2](#), [7](#), [15](#)
- [36] Anthony Simeonov, Yilun Du, Lin Yen-Chen, Alberto Rodriguez, Leslie Pack Kaelbling, Tomas Lozano-Perez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. *arXiv preprint arXiv:2211.09786*, 2022. [2](#)
- [37] Michael Strecke and Joerg Stueckler. EM-Fusion: Dynamic object-level SLAM with probabilistic data association. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2019. doi: 10.1109/iccv.2019.00596. [2](#), [9](#), [10](#)
- [38] Edgar Sucar, Kentaro Wada, and Andrew Davison. NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 949–958. IEEE, 2020. [1](#), [2](#)
- [39] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. [1](#), [2](#), [3](#)
- [40] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2295–2302, 2016. doi: 10.1109/ICRA.2016.7487378. [1](#)
- [41] Jingwen Wang, Martin Rünz, and Lourdes Agapito. DSP-SLAM: Object oriented SLAM with deep shape priors. In *2021 International Conference on 3D Vision (3DV)*, pages 1362–1371. IEEE, 2021. [1](#), [2](#)
- [42] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. [7](#)
- [43] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic SLAM. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5231–5237. IEEE, 2019. [1](#), [2](#)
- [44] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-D object SLAM. *IEEE Transactions on Robotics*, 35(4):925–938, 2019. [1](#), [2](#), [9](#), [10](#)
- [45] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In *ICRA*, 2022. [2](#)
- [46] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11776–11785, 2019. [2](#)
- [47] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. [9](#)
- [48] Minghan Zhu, Maani Ghaffari, and Huei Peng. Correspondence-free point cloud registration with so (3)-equivariant implicit shape representations. In *Conference on Robot Learning*, pages 1412–1422. PMLR, 2022. [2](#)
- [49] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-SLAM: Neural implicit scalable encoding for SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. [1](#), [2](#), [3](#)

VII. APPENDIX

Here, we provide extra details on our experiment setup and mapping procedures.

A. Experimental Details

Model Details. We provide the network architecture of our encoder and decoder in Table. IV and Table. V, respectively, which are adopted from Neural Descriptor Fields [35] using the VNNLinear, VNNResnetBlock, VNLeakyReLU blocks introduced in Vector Neurons [8]. Here, z_{dim} refers to the size of the NeuSE latent code, $\mathbf{z} \in \mathbb{R}^{z_{dim} \times 3}$.

Training Details. During training, we randomly draw 500 points from the observed partial point clouds for each sample to be fed into the encoder network. For training with L_{occ} , the query point size is 750 consisting of half object points and half off-the-object points from the given model.

We set the dimension of the latent code \mathbf{z} to be $z_{dim} = 512$ and the weight coefficients $(\beta_1, \beta_2) = (0.1, 0.1)$ for unambiguous objects and $(\beta_1, \beta_2) = (1, 0.1)$ for ambiguous objects so as to balance the order of magnitude difference among L_{occ} , $L_{transform}$, and L_{shape} (with L_{occ} as the reference). In terms of our choice of the latent code dimension, we further find that, as opposed to training the three losses jointly using a single 512-dimensional latent code, we may also enforce SE(3)-equivariance and cross-viewing-angle shape consistency separately on two lower dimensional latent codes/networks. Here, when applying $L_{occ} + L_{transform}$ and $L_{occ} + L_{shape}$ individually on two network models each with a latent size of 128, we obtain almost on-par transform and shape characterization power from the combination of two 128-dimensional latent codes compared to that of the vanilla 512-dimensional code. Hence, this can serve as a memory-efficient alternative to our original training approach for more lightweight training and memory-critical application scenarios.

B. Reconstruction and Update of the Object-centric Map

In this section, we elaborate on our choices and procedures in building and maintaining the object-centric map, which we adopt to deal with noisy real-world data.

Our proposed approach depends on correct object segmentation masks to produce effective latent codes for objects. To avoid potential failures from false object latent codes, considering the uncertainty of off-the-shelf object detectors and depth cameras, we only initialize a new object instance if it has been recognized robustly by the depth camera and the object detector, e.g., an object close enough to the camera with an abundant number of points in the observed point cloud. Here in our experiment for object instantiation, we only consider objects that are within 2m away from the depth camera and with their pixel-level segmentation mask size above 4000. Ultimately, a new object is instantiated after it has been regarded as a “new” object three times in a row during data association.

For map maintenance and update, we conduct the removal of “residual” object instances based on the bounding box of each full object reconstruction. This ensures less redundant

VNLinear(128,256)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
VNLinear(256,128)
Meanpool
VNLinear(128, z_{dim})
$\mathbf{z} \leftarrow \text{Encode}$

TABLE IV: Encoder architecture.

VNLinear(z_{dim}, z_{dim})
Linear($2 * z_{dim} + 1, 128$)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
ResnetBlockFC(128)
Linear(128,1)
Sigmoid

TABLE V: Decoder architecture.

object instantiation from partial observations and no overlapping object reconstructions for detected changed objects whose new positions were previously occupied. Thanks to NeuSE, we can always get a reasonable shape prediction out of partial observations, e.g., an upright bottle with its bottom half obscured. We are, therefore, able to obtain the 3D bounding box of each observed object and conduct the object removal procedure as follows: (1) If a small object’s bounding box has a high overlap with a large one (0.95 in our case), this small object is deemed as a partial instance belonging to the large one and will be removed. (2) A changed object’s bounding box intersects (we set it as 20% of its bounding box volume) with an older object, meaning the older object should no longer be in its original place. We hence remove the older object from the map and update the changed object to its new position. In this way, we are able to maintain a consistent object map while avoiding overlapping reconstructions such as the artifacts shown in Fig. 12(d).