Reconstructing Interactive 3D Scenes
by Panoptic Mapping and CAD Model Alignments

Muzhi Han* Zeyu Zhang* Ziyuan Jiao Xu Xie Yixin Zhu Song-Chun Zhu Hangxin Liu

Abstract—In this paper, we rethink the problem of scene reconstruction from an embodied agent’s perspective: While the classic view focuses on the reconstruction accuracy, our new perspective emphasizes the underlying functions and constraints such that the reconstructed scenes provide actionable information for simulating interactions with agents. Here, we address this challenging problem by reconstructing an interactive scene using RGB-D data stream, which captures (i) the semantics and geometry of objects and layouts by a 3D volumetric panoptic mapping module, and (ii) object affordance and contextual relations by reasoning over physical common sense among objects, organized by a graph-based scene representation. Crucially, this reconstructed scene replaces the object meshes in the dense panoptic map with part-based articulated CAD models for finer-grained robot interactions. In the experiments, we demonstrate that (i) our panoptic mapping module outperforms previous state-of-the-art methods, (ii) a high-performant physical reasoning procedure that matches, aligns, and replaces objects’ meshes with best-fitted CAD models, and (iii) reconstructed scenes are physically plausible and naturally afford actionable interactions; without any manual labeling, they are seamlessly imported to ROS-based simulators and virtual environments for complex robot task executions.1

I. I NTRODUCTION

Perception of the human-made scenes and the objects within inevitably leads to the course of actions [1, 2]: such a task-oriented view [3, 4] is the basis for a robot to interact with the environment and accomplish complex tasks. In stark contrast, such a crucial perspective is largely missing in the robot mapping and scene reconstruction literature: Prevailing semantic mapping or Simultaneous Localization and Mapping (SLAM) methods often produce a metric map of the scene with semantic or instance annotations; they only emphasize mapping accuracy but omit the essence of robot task execution—actions that a semantic entity could afford and associated physical constraints embedded among entities.

Such a lack of the scene’s functional representation leads to a gap between the reconstructed semantic scenes and Task and Motion Planning (TAMP), which prevents a robot from directly interacting with the reconstructed scenes to accomplish complex tasks. Take the reconstructed scene in Fig. 1 as the example, wherein the robot is tasked to pick up a frozen meal from the fridge, microwave and serve it. To properly plan and execute inside the reconstructed scene, a robot ought to acquire (i) semantics and geometry of objects (e.g., this piece of point cloud is a fridge), (ii) actions an object affords (e.g., a fridge can be open), and (iii) constraints among these entities (e.g., no objects should float in the air).

Although modern semantic mapping and SLAM methods can partially address (i) [5, 6], existing solutions for (ii) [4, 7, 8] and (iii) [9–14] have not yet been fully integrated into a robot scene reconstruction framework, resulting in non-interactive reconstructed scenes. This deficiency precludes the feasibility of directly applying TAMP on the reconstructed scenes either using traditional [15, 16] or learning-based [17, 18] methods; the robot can hardly verify whether its plan is valid or the potential outcomes of its actions are satisfied before executing in the physical world.

Although researchers have attempted to devise manual pipelines (e.g., iGibson [19], SAPIEN [20]) to either convert the reconstructed real-world scenes or directly build virtual environments from scratch, creating such simulation environments is a non-trivial and time-consuming task. The simulated environment should be sufficiently similar to the reality, and the objects to be interacted with should afford sufficiently similar functionality. Only by satisfying the above conditions could the outcomes of interactions in simulation be similar to those in the physical world. Due to the enormous workload to create/convert each scene, the number of available scenes to date is still quite limited. A challenge naturally arises: Can we reconstruct a scene that can be automatically imported into various simulators for interactions and task executions?

In this paper, we propose a new task of reconstructing functionally equivalent and interactive scenes, capable of being directly imported into simulators for robot training.

1 The code is available at https://github.com/hmz-15/Interactive-Scene-Reconstruction

* Muzhi Han and Zeyu Zhang contributed equally to this work.

1 The code is available at https://github.com/hmz-15/Interactive-Scene-Reconstruction

* Muzhi Han and Zeyu Zhang contributed equally to this work.

UCLA Center for Vision, Cognition, Learning, and Autonomy (VCLA) at the Statistics Department. Emails: {muzhihan, zeyuzhang, zyi, xie, yixin.zhu, lx.liu}@ucla.edu, sczhu@stat.ucla.edu.

The work reported herein was supported by ONR N00014-19-1-2153, ONR MURI N00014-16-1-2007, and DARPA XAI N66001-17-2-4029.

Fig. 1: The reconstruction of an interactive 3D scene. (a) A contact graph is constructed by the supporting relations that emerged from (b) panoptic mapping. By reasoning their affordance, functional objects within the scene are matched and aligned with part-based interactive CAD models. (c) The reconstructed scene enables a robot simulates its task execution with comparable outcomes in the physical world.
and testing of complex task execution. We argue that a scene’s functionality is composed of the functions afforded by objects within the scene. Therefore, the essence of our scene reconstruction lies in defining functionally equivalent objects, which should preserve four characteristics with decreasing importance: (i) its semantic class and spatial relations with nearby objects, (ii) its affordance, e.g., what interactions it offers, (iii) a similar geometry in terms of size and shape, and (iv) a similar appearance.

Existing approaches oftentimes represent reconstructed semantic scene and its entities as sparse landmarks [21, 22], surfels [5, 23], or volumetric voxels [24, 25]. However, these representations are inadequate to serve as a functional representation of the scene and its entities: They merely provide occupancy information (i.e., where the fridge is) without any actionable information for robot interactions or planning (e.g., whether or how the fridge can be open).

To address the above issues, we devise three primary components in our system; see an illustration in Fig. 2:

(A) **A robust 3D volumetric panoptic mapping module**, detailed in Section III, accurately segments and reconstructs 3D objects and layouts in clustered scenes even with noisy per-frame image segmentation results. The term “panoptic,” introduced in [26], refers to jointly segmenting stuff and things. In this paper, we regard objects as things and layout as stuff. Our system produces a volumetric panoptic map using a novel per-frame panoptic fusion and a customized data fusion procedure; see examples in Fig. 1b and Fig. 2a.

(B) **A physical common sense reasoning module**, detailed in Section IV, replaces object meshes obtained from the panoptic map with interactive rigid or articulated CAD models. This step is achieved by a ranking-based CAD data fusion procedure; see examples in Fig. 1b and Fig. 2a.

(C) **A graphical representation, contact graph cg**, (Fig. 1a, Fig. 2c, and Section II) is built and maintained simultaneously, in which the nodes of a cg represent objects and layouts, and the edges of a cg denote the support and proximal relations. We further develop an interface to convert a cg to a Unified Robot Description Format (URDF) such that the reconstructed functionally equivalent scene (see Fig. 1C) can be directly imported into simulators for robot interactions and task executions; see Section V for experimental results.

**Related Work:** Existing approaches to generate simulated interactive environments fall into three categories: (i) **manual efforts**, such as those in Gazebo [27] and V-REP [28] for robotics, AI2THOR [29] and Gibson [30] for embodied AI, and iGibson [19], SAPIEN [20], and VR-Gym [31] with part-based articulated objects (e.g., a cabinet with a door); (ii) **scene synthesis** that produces a massive amounts scenes with the help of CAD databases [32–34]; (iii) **large-scale scene dataset** with aligned CAD models, such as SUNCG [35] and 3D-FRONT [36]. However, without tedious manual work, all of these prior approaches fail to replicate a real scene in simulation with diverse interactions.

Modern semantic mapping [6, 24, 37] and object SLAM [22, 25] methods can effectively reconstruct an indoor scene at an object-level. Physical cues, such as support and collision, has been further integrated to estimate and refine the object pose [38–40]. In parallel, computer vision algorithms predict 3D instance segmentation in densely reconstructed scenes [41, 42], and then fit CAD models by crowdsourcing [43] or by computing the correspondences between the reconstructed scenes and CAD models [44, 45]. However, the above work fails to go beyond semantics to (i) capture the interactive nature of the objects, or (ii) meaningfully represent a physically plausible scene. As such, the reconstructed scenes still fail to be imported into simulators to afford robot interactions and task executions.

Constructing a proper scene or a map representation remains an open problem [46]. Typical semantic mapping and SLAM methods only output a flat representation, difficult to store or process high-level semantics for robot interactions and task executions. Meanwhile, graph-based representations, e.g., scene grammar [11, 13, 14, 34, 47, 48] and 3D scene graph [49–51], provide structural and contextual information. In particular, Rosinol et al. [51] also incorporate actionable information for robot navigation tasks. Our work devise a contact graph with supporting and proximal relations, which imposes kinematic constraints for more complex robot manipulation.

**II. Contact-based Scene Representation**

We devise a graph-based representation, contact graph cg, to represent a 3D indoor scene. Formally, a contact graph cg = (pt, E) contains (i) a parse tree (pt) that captures the hierarchical relations among the scene entities [47], and (ii) the proximal relations E among entities represented by undirected edges; see an example of pt in Fig. 1a.

**A. Representation**

**Scene Parse Tree** pt = (V, S) has been used to represent the hierarchical decompositional relations (i.e., the edge set S) among entities (i.e., the node set V) in various task domains, including 2D images and 3D scenes [11, 13, 14, 33, 34, 48, 52], videos and activities [4, 8, 53], robot manipulations [54–58], and theory of mind [59]. In this paper, we adopt pt to represent supporting relations among entities, dynamically built and maintained during the reconstruction; for instance in Fig. 1a, the cabinet is the parent node of the microwave. Supporting relation is quintessential in scene understanding with physical common sense as it reflects the omnipresent physical plausibility; i.e., if the cabinet were moved, the microwave would move together with it or fall onto the ground. This counterfactual perspective goes beyond occupancy information (i.e., the physical location of an object); in effect, it further provides actionable information and the potential outcome of actions for robot interactions and task executions in the scene.

**Scene Entity Nodes** V = {v_x} ∪ V_L ∪ V_R ∪ V_A include: (i) the scene node v_x, seving as the root of pt, (ii) layout node set V_L, including floor, ceiling, and the wall that bound the 3D scene, (iii) rigid object set V_R, wherein each object has no articulated part (e.g., a table), and (iv) articulated...
object set \( V^A \), wherein each object has articulated parts to be interacted for various robot tasks (e.g., fridge, microwave).

Each non-root node \( v_i = (o_i, c_i, M_i, B_i(p_i, q_i, s_i), \Pi_i) \) encodes a unique instance label \( o_i \), a semantic label \( c_i \), a full geometry model \( M_i \) (a triangular mesh or a CAD model), a 3D bounding box \( B_i \) (parameterized by its position \( p_i \), orientation \( q_i \), and size \( s_i \), all in \( \mathbb{R}^3 \)), and a set of surface planes \( \Pi_i = \{ \pi^k, k = 1 \cdots |\Pi_i| \} \), where \( \pi^k \) is a homogeneous vector \( [n^k_T, d^k] \in \mathbb{R}^4 \) in the projective space [60] with unit plane normal vector \( n^k \), and any point \( v \in \mathbb{R}^3 \) on the plane satisfies a constraint: \( n^k_T \cdot v + d^k = 0 \).

**Supporting Relations** \( S \) is the set of directed edges in \( pt \) from parent nodes to their child nodes. Each edge \( s_{p,c} \in S \) imposes physical common sense between the parent node \( v_p \) and the child node \( v_c \). These constraints are necessary to ensure that \( v_p \) supports \( v_c \) in a physically plausible fashion:

(i) **Geometrical plausibility:** \( v_p \) should have a plane \( \pi^p = [n^p_T, d^p] \) with \( n^p \) being opposite to the gravity direction, whereas bottom surface of \( v_c \) should contact the top of \( \pi^p \):

\[
\exists n^p \in \Pi_p, n^p_T \cdot g \leq a_{th}, \quad \text{s.t.} \quad D(v_c, \pi^p) = p^c_T - (d^p_T + s^c_T/2) = 0, \quad (1)
\]

where \( g \) is the unit vector along the gravity direction, \( a_{th} = -0.9 \) is a tolerance coefficient, \( d^p_T \) is the offset of the \( v_p \)'s supporting plane, and \( p^c_T \) and \( s^c_T \) denote the position and size of the \( v_c \)'s 3D bounding box along the gravity direction.

(ii) **Sufficient contact area for stable support:** Formally,

\[
A(v_p, v_c) = A(v_p \cap v_c)/A(v_c) \geq b_{th}, \quad (2)
\]

where \( A(v_c) \) is the bottom surface of the \( v_c \)'s 3D bounding box, and \( A(v_p \cap v_c) \) is the area of the overlapping rectangle containing the mesh vertices of \( v_p \) near \( \pi^p \) within \( v_c \)'s 3D bounding box. We set threshold \( b_{th} = 0.5 \) for a stable support.

**Proximal Relations** \( E \) introduce links among entities in the \( pt \). They impose additional constraints by modeling spatial relations between two non-supporting but physically nearby objects \( v_1 \) and \( v_2 \): Their meshes should not penetrate with each other, i.e., \( \text{Vol}(M_1 \cap M_2) = 0 \). Note that the constraint only exists between two objects with overlapping 3D bounding boxes, i.e., when \( \text{Vol}(B_1 \cap B_2) > 0 \).

B. Constructing Contact Graph

Each node \( v_x \) in \( cg \) is constructed from a scene entity \( x \) in the panoramic map (see Section III) by: (i) acquiring its \( a_x, c_x, M_x, B_x(p_x, q_x, s_x) \), (ii) extracting surface planes \( \Pi_x \) by iteratively applying RANSAC [61] and removing plane inliers, and (iii) assigning \( x \) as \( v_x \) in \( cg \).

Given a set of nodes constructed on-the-fly, we apply a bottom-up process to build up \( cg \) by detecting supporting relations among the entities. Specifically, given an entity \( v_c \), we consider all entities \( \{v_i\} \) whose 3D bounding boxes are spatially below it and have proper supporting planes \( \pi^k \) based on Eq. (1). The most likely supporting relation is chosen by maximizing the following score function:

\[
S(v_c, v_i, \pi^k) = \{1 - \min [1, \|D(v_c, \pi^k)\|] \} \times A(v_i, v_c), \quad (3)
\]

where the first term indicates the alignment between the \( v_c \)'s bottom surface and the \( v_i \)'s supporting planes, and the second term reflects an effective supporting area, both normalized to \([0,1] \). \( B_i \) is further refined (see Eq. (1)) as it was computed based on incomplete object meshes. Meanwhile, the proximal relations are assembled by objects’ pairwise comparison. At length, the \( cg \) of the scene is constructed based on the identified entities and their relations and grows on-the-fly.

III. ROBUST PANOPTIC MAPPING

Robust and accurate mapping of scene entities within clustered environments is essential for constructing a \( cg \) and serving downstream tasks. Below, we describe our robust panoramic mapping module to generate volumetric object and layout segments in the form of meshes from RGB-D streams; see the pipeline in Fig. 2A. We follow the framework
proposed in [24] and only highlight crucial technical modifications below. The experiments demonstrate that our modifications significantly improve system performance.

**Per-frame Segmentation:** We combine the segmentation of both RGB and depth for performance improvement as in [24]. However, instead of merely labeling the depth segments with semantic-instance masks, we bilaterally fuse panoramic masks and geometric segments to output point cloud segments with both semantic and instance labels. We further perform an outlier removal for each object entity; far-away segments are removed and assigned to the scene background.

This modification significantly improves the noisy per-frame segmentation; see Fig. 2a. In this example, fusing RGB and depth segments mutually improves the segments if they were obtained by each alone. The fusion (i) correctly segments the keyboard and divides the two monitors when depth segments fail, and (ii) geometrically refines the noisy panoramic mask of the chair to exclude the far-away ground.

**Data Fusion:** Compared to [24], we introduce two notable enhancements in data fusion. First, we use a triplet count \( \Phi(l, c, o) \) to record the frequency that an instance label \( o \), a semantic label \( c \), and a geometric label \( l \) associated with the same point cloud segment; it is incrementally updated: \( \Phi(l, c, o) = \Phi(l, c, o) + 1 \). This modification improves consistency in semantic-instance fusion. Second, in addition to merging two geometric labels if they share voxels over a certain ratio, we also regulate two instance labels if the duration of association with a common geometric label exceeds a threshold. We further estimate a gravity-aligned, 3D-oriented bounding box for each object mesh [62]. In sum, our system simultaneously and comprehensively outputs a set of scene entities with their instance labels, semantic labels, 3D bounding boxes, and reconstructed meshes.

**Implementation and Evaluation:** We use an off-the-shelf panoramic segmentation model [63] pre-trained on the COCO panoramic class [64] for RGB images and a geometric segmentation method [65] for depth images. We compare our panoramic mapping module with the original Voxblox++ [24] on 8 sequences in the SceneNN dataset [66]. Our evaluation includes four criteria: (i) panoramic quality (PQ) [6, 26], (ii) segmentation quality (SQ), (iii) recognition quality (RQ) of 3D panoramic mapping on 8 thing classes and 2 stuff classes, and (iv) the mean average precision (mAP) computed using an intersection of union (IoU) with a threshold of 0.5 for 3D oriented bounding box estimation on thing classes. Since the supporting relations in \( cg \) could further refine the 3D bounding boxes (see Section II-B), we also include mAP in (c).

Table 1 tabulates the class-averaged results, showing that our method consistently outperforms the baseline in both 3D panoramic mapping and 3D bounding box estimation; see Fig. 5b for some qualitative results. In general, refining objects’ 3D bounding boxes with supporting relations introduces significant improvement in accuracy.

### IV. PHYSICAL REASONING FOR CAD ALIGNMENTS

Due to occlusion or limited camera view, the reconstructed meshes of the scene are oftentimes incomplete. As such, the segmented object meshes are incomplete and non-interactive before recovering them as full 3D models; see examples in

<table>
<thead>
<tr>
<th>Sequence ID</th>
<th>PQ</th>
<th>SQ</th>
<th>RQ</th>
<th>mAP_c</th>
<th>mAP_v</th>
<th>PQ</th>
<th>SQ</th>
<th>RQ</th>
<th>mAP_v</th>
</tr>
</thead>
<tbody>
<tr>
<td>061</td>
<td>43.0</td>
<td>52.0</td>
<td>46.3</td>
<td>33.6</td>
<td>33.6</td>
<td>25.7</td>
<td>53.1</td>
<td>32.2</td>
<td>8.9</td>
</tr>
<tr>
<td>086</td>
<td>27.3</td>
<td>39.6</td>
<td>34.6</td>
<td>33.8</td>
<td>33.8</td>
<td>19.4</td>
<td>32.9</td>
<td>25.2</td>
<td>7.9</td>
</tr>
<tr>
<td>096</td>
<td>12.5</td>
<td>21.4</td>
<td>14.6</td>
<td>14.6</td>
<td>14.6</td>
<td>7.3</td>
<td>11.0</td>
<td>8.3</td>
<td>14.6</td>
</tr>
<tr>
<td>223</td>
<td>49.5</td>
<td>60.2</td>
<td>63.3</td>
<td>52.4</td>
<td>52.4</td>
<td>21.7</td>
<td>40.2</td>
<td>26.7</td>
<td>61.4</td>
</tr>
<tr>
<td>225</td>
<td>35.4</td>
<td>46.9</td>
<td>44.8</td>
<td>31.5</td>
<td>31.5</td>
<td>19.7</td>
<td>43.6</td>
<td>29.4</td>
<td>11.2</td>
</tr>
<tr>
<td>231</td>
<td>37.8</td>
<td>45.9</td>
<td>45.4</td>
<td>29.2</td>
<td>29.2</td>
<td>17.9</td>
<td>30.4</td>
<td>22.1</td>
<td>19.4</td>
</tr>
<tr>
<td>249</td>
<td>24.4</td>
<td>33.8</td>
<td>34.4</td>
<td>48.9</td>
<td>48.9</td>
<td>23.4</td>
<td>36.4</td>
<td>30.6</td>
<td>48.5</td>
</tr>
<tr>
<td>322</td>
<td>68.4</td>
<td>71.1</td>
<td>80.0</td>
<td>53.3</td>
<td>83.3</td>
<td>43.6</td>
<td>64.6</td>
<td>52.9</td>
<td>25.0</td>
</tr>
</tbody>
</table>

**A. CAD Pre-processing**

We collected a CAD database consisting of both rigid and articulated CAD models, organized by semantic classes. The rigid CAD models are obtained from ShapeNetSem [67], whereas articulated parts are first assembled and then properly transformed into one model. Each CAD is transformed to have its origin and axes aligned with its canonical pose. Fig. 2B shows some instances of CAD models in the database. Similar to a segmented object entity, a CAD model \( y \) is parameterized by \( o_y, c_y, M_y, B_y(p_y, q_y, s_y) \), and \( \Pi_y \).

**B. Ranking-based CAD Matching**

Take the chair in Fig. 2b as an example: Given a segmented object entity \( x \), the algorithm retrieves all CAD models in the same semantic category (i.e., chair) from the CAD database to best fit \( x \)’s geometric information. Since the exact orientation of \( x \) is unknown, we uniformly discretize the orientation space into 24 potential orientations. For each rotated CAD model \( y \) that aligned to one of the 24 orientations, the algorithm computes a matching distance:

\[
D(x, y) = \omega_1 \cdot d_s(x, y) + \omega_2 \cdot d_n(x, y) + \omega_3 \cdot d_b(y),
\]

where \( \omega_1 = \omega_2 = 1.0 \) and \( \omega_3 = 0.2 \) are the weights of three terms, set empirically. We detail these terms below.

(i) \( d_s \) matches the relative sizes of 3D bounding boxes:

\[
d_s(x, y) = \frac{s_x}{\|s_x\|_2} - \frac{s_y}{\|s_y\|_2}.
\]

(ii) \( d_n \) penalizes the misalignment between their surface planes in terms plane normal and relative distance:

\[
d_n(x, y) = \min_{\tau \in \Pi_x} \sum_{\pi \in \Pi_x} \left[ \frac{d(T_x^\top \pi_i) - d(f_{11}(\pi_i))}{\|s_y\|_2} \right] + 1 - n(\pi_i)^T n(f_{11}(\pi_i)),
\]

where \( T_x \) denotes the homogeneous transformation matrix from the map frame on the ground to the frame of the bounding box \( B_x \), \( d(\cdot) \) and \( n(\cdot) \) denote the offset and normal vector of a plane, and \( f_{11}: \Pi_x \rightarrow \Pi_y \) is a bijection function denoting the assignment of feature planes between \( x \) and \( y \).

Note that \( f_{11} \) is also constrained to preserve supporting planes
Fig. 3: Examples of matching and aligning CAD candidates to (a) an input object mesh. (b) All CAD models within the same semantic class as the input object are retrieved for matching. Matching Error (ME) reflects both the similarity in shapes and the proximity in orientations. After selecting the CAD candidates with smallest MEs, (c) a fine-grained CAD alignment process selects the best CAD model with a proper transformation based on Alignment Error (AE).

as defined in Eq. (1). As computing $d_e$ involves solving an optimal assignment problem, we adopt a variant of the Hungarian algorithm [68] to identify the best $f_{ij}$.

(iii) $d_b(y)$ is a bias term that adjusts the overall matching error for less preferable CAD candidates:

$$d_b(y) = 1 + g^T \cdot z(y),$$  

(7)

where $z(y)$ denotes the up-direction of the CAD model in the oriented CAD frame, and $g$ is a unit vector along the gravity direction. In general, we prefer CAD candidates that stand upright to those leaning aside or upside down.

Fig. 3b illustrates the matching process. Empirically, we observe that the discarded CAD candidates of “chair” and “table” due to large Matching Error (ME) are indeed more visually distinct from the input object meshes. Moreover, the “fridge” model with a wrong orientation has a much larger ME and is thus discarded. These results demonstrate that our ranking-based matching process can select visually more similar CAD models with the correct orientation. Our system maintains the top 10 oriented CAD candidates with the lowest ME for the fine-grained alignment in the next stage.

C. Optimization-based CAD Alignment

Given a shortlist of CAD candidates, the overarching goal of this step to find an accurate transformation (instead of 24 discretized orientations) that aligns a given CAD candidate $y$ to the original object entity $x$, achieved by estimating a homogeneous transformation matrix between $x$ and $y$:

$$T = \begin{bmatrix} \alpha R & p \\ 0^T & 1 \end{bmatrix}, \text{ s.t. } \min_T J(x, T \circ y),$$  

(8)

where $\circ$ denotes the transformation of a CAD candidate $y$, $J$ is an alignment error function, $\alpha$ is a scaling factor,

$$R = \text{Rot}(z, \theta)$$ is a rotation matrix that only considers the yaw angle under the gravity-aligned assumption, and $p$ is a translation. This translation is subject to the following constraint: $p^T = -d^x + \alpha \cdot s^y / 2$, as the aligned CAD candidate is supported by a supporting plane $\pi = [n^x, d^x]$.

The objective function $J$ can be written in a least squares form and minimized by the Levenberg–Marquardt [69] method:

$$J = e_b^T \Sigma_b e_b + e_p^T \Sigma_p e_p,$$  

(9)

where $e_b$ is the 3D bounding box error, $e_p$ the plane alignment error, and $\Sigma_b, \Sigma_p$ the error covariance matrices of the error terms. Specifically: (i) $e_b$ aligns the height of the two 3D bounding boxes while constraining the ground-aligned rectangle of the transformed $B_y$ inside that of $B_y$:

$$e_b = ([A(T \circ y)] - A(x, T \circ y), \alpha \cdot s^y - s^y)^T,$$  

(10)

and (ii) $e_p$ aligns all the matched feature planes as:

$$e_p = [\Delta \pi, \ldots, \Delta \pi_{|B_y|}]^T,$$

$$\Delta \pi = [-d(\pi, i) + d(T^{-T} \cdot f_s(\pi, i)), 1 - n(\pi, i)^T \cdot n(T^{-T} \cdot f_s(\pi, i))].$$  

(11)

We evaluate each aligned CAD candidate by computing an Alignment Error (AE), the root mean square distance between the object mesh vertices and the closest points on aligned CAD candidate; Fig. 3c shows both qualitative and quantitative results. The CAD candidate with the smallest AE will be selected, whereas others are potential substitutions if the selected CADs violate physical constraints, detailed next.

D. Global Physical Violation Check

Given a shortlist of matched and aligned CAD candidates, we validate supporting relations and proximal relations; see Fig. 4 for qualitative results. Specifically, for an object node $v_p$ and its object entity $x$, we discard a CAD candidate $y$ if it fails to satisfy Eq. (2) with any supporting child $v_c$ of $v_p$. We also check the proximal constraint by first discarding CAD candidates that collide with the layout.
entities, and then jointly selecting CAD candidates for each object entity to guarantee the object-object non-collision. The joint selection problem can be formulated as a constraint satisfaction problem. Starting with a CAD candidate with the minimum alignment error for each object entity, we adopt the min-conflict algorithm [70] to obtain a global solution.

V. EXPERIMENTS AND RESULTS

We perform scene reconstruction experiments using RGB-D sequences in the SceneNN dataset [66] and import the results into various simulators for interaction; see Fig. 5. Compared to the ground-truth segmentation, our panoptic mapping system accurately recognizes and segments scene entities (Fig. 5b). Such an accurate mapping provides the basis for high-level physical reasoning to replace incomplete meshes with CAD models, resulting in a high-quality, functionally equivalent, interactive scene reconstruction, as shown in Fig. 5c. Note that our system’s performance could be further improved as we only utilize pre-trained models in the mapping procedure without fine-tuning. The run-time for converting a 3D panoptic map into an interactive scene varies from 30 seconds to several minutes, depending on the number and categories of functional objects involved.

The reconstructed scene $c_g$ can be readily converted into a URDF and be imported into robot simulators. While it is straightforward to immigrate scene entities in $c_g$ to links and joints in the kinematic tree, supporting edges are altered to fixed/floating joints based on the semantics of the scene entity pairs (e.g., a cup is connected to a table using a floating joint as it can be freely manipulated). Fig. 5c shows the reconstructed scenes in the ROS environment, which subsequently connects the reconstructed scenes and robot TAMP; see Fig. 5d. Fig. 5e demonstrates that the reconstructed scenes can be loaded into the VR environment [31] for interactions with both virtual agents and human users, which opens a new avenue for future studies.

VI. CONCLUSIONS

We proposed a new task of reconstructing interactive scenes that captures the semantic and associated actionable information of objects in a scene, instead of purely focusing on geometric reconstruction accuracy. We solved this new task by combining (i) a novel robust panoptic mapping that segments individuals objects and layouts, and (ii) a physical reasoning process to replace incomplete objects meshes with part-based CAD models, resulting in physically plausible and interactive scenes. We validated the capability of our system with both qualitative and quantitative results. Finally, we showed that various simulators (e.g., ROS, VR environments) can seamlessly import the reconstructed scene to facilitate researches in robot TAMP and embodied AI.

This work also motivates three new research questions worth investigating in the future: (i) To sufficiently plan robot tasks, how well should the CAD models replicate the real objects? (ii) Although the proposed system can filter out dynamic entities based on their semantic segmentation (e.g., humans) and a better data association can handle semi-dynamic objects, how could we incorporate the causal relations between environmental changes and human activities? (iii) Although the effects of acting in a sequential task could be updated as the kinematic information in $c_g$, recognizing these effects in physical world introduces extra challenges.