REACT3D: Recovering Articulations for Interactive Physical 3D Scenes

Zhao Huang[1], Boyang Sun[1], Alexandros Delitzas[1,2], Jiaqi Chen[1], and Marc Pollefeys[1,3]

***Abstract*—Interactive 3D scenes are increasingly vital for embodied intelligence, yet existing datasets remain limited due to the labor-intensive process of annotating part segmentation, kinematic types, and motion trajectories. We present REACT3D, a scalable zero-shot framework that converts static 3D scenes into simulation-ready interactive replicas with consistent geometry, enabling direct use in diverse downstream tasks. Our contributions include: (i) openable-object detection and segmentation to extract candidate movable parts from static scenes, (ii) articulation estimation that infers joint types and motion parameters, (iii) hidden-geometry completion followed by interactive object assembly, and (iv) interactive scene integration in widely supported formats to ensure compatibility with standard simulation platforms. We achieve state-of-the-art performance on detection/segmentation and articulation metrics across diverse indoor scenes, demonstrating the effectiveness of our framework and providing a practical foundation for scalable interactive scene generation, thereby lowering the barrier to large-scale research on articulated scene understanding.**

***Index Terms*—Semantic scene understanding, object detection, segmentation and categorization, RGB-D perception.**

I. Introduction

HIGH-FIDELITY, interactive 3D assets are critical for a wide range of applications. They enable immersive experiences in virtual and mixed reality, support realistic content creation for gaming and film production, and facilitate the development of autonomous robotic systems by serving as training environments for navigation and manipulation tasks [1]. These applications demand large-scale 3D datasets that offer both photorealistic rendering and physically plausible interactions, such as object picking, placing, or articulating structures like doors and drawers. Automating the generation of such datasets is essential to scale up these efforts effectively.

Significant effort has been dedicated to the automatic generation of 3D assets. Early works primarily focused on the creation of *static* 3D assets, often converting real-world scans or image sequences into accurate 3D representations such as meshes or point clouds [2]. These efforts have led to the release of several high-quality static 3D datasets. More recently, the rapid advancement of generative models has enabled the synthesis of large-scale static scenes with diverse modalities and formats [3]. As a result, the generation of static 3D scenes, whether through real-world reconstruction or generative modeling, has become increasingly mature.

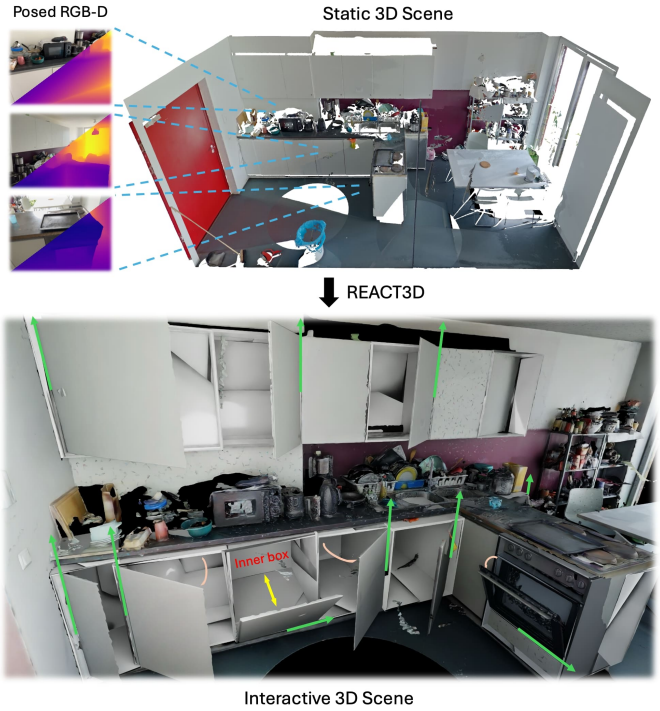


Fig. 1. **REACT3D** transforms static 3D scenes into interactive scenes in a zero-shot manner. The generated interactive scenes are spatially aligned with the static input and preserve the original geometry and appearance. Our results are readily compatible with multiple simulation platforms, supporting diverse downstream tasks such as robotic perception, interaction, and embodied intelligence.

In parallel, the research community has begun to explore the generation of *dynamic* scenes, either by reconstructing real-world interaction sequences [4] or by manually designing 3D environments with humans in the loop. However, despite promising progress, the scalability and quality of interactive scene generation still lag behind those of static scene generation.

In this work, we are motivated by the growing need for large-scale, interactive 3D environments to support embodied AI research. We propose a novel, application-driven framework for converting high-quality static scenes into physics-aware, interactive environments (Fig. 1). Our system leverages recent advances in vision foundation models to enable zero-shot transfer from standard 3D scene formats (e.g., `.ply`,

This work was supported by the Swiss National Science Foundation Advanced Grant 216260: "Beyond Frozen Worlds: Capturing Functional 3D Digital Twins from the Real World", and the European Union's Horizon Europe research and innovation programme under grant agreement number 101214398 (ELLIOT). Alexandros Delitzas is supported by the Max Planck ETH Center for Learning Systems (CLS).

[1]All authors are with ETH Zurich, 8092 Zurich, Switzerland zhahuang@ethz.ch; boyang.sun@inf.ethz.ch; alexandros.delitzas@inf.ethz.ch; jiaqi.chen@inf.ethz.ch; marc.pollefeys@inf.ethz.ch

[2]Alexandros Delitzas is also with Max Planck Institute for Informatics, 66123 Saarbruecken, Germany alexandros.delitzas@mpi-inf.mpg.de

[3]Marc Pollefeys is also with Microsoft Spatial AI Lab, 8038 Zurich, Switzerland mapoll@microsoft.com

Project page at https://react3d.github.io.

Digital Object Identifier (DOI): 10.1109/LRA.2026.3674028

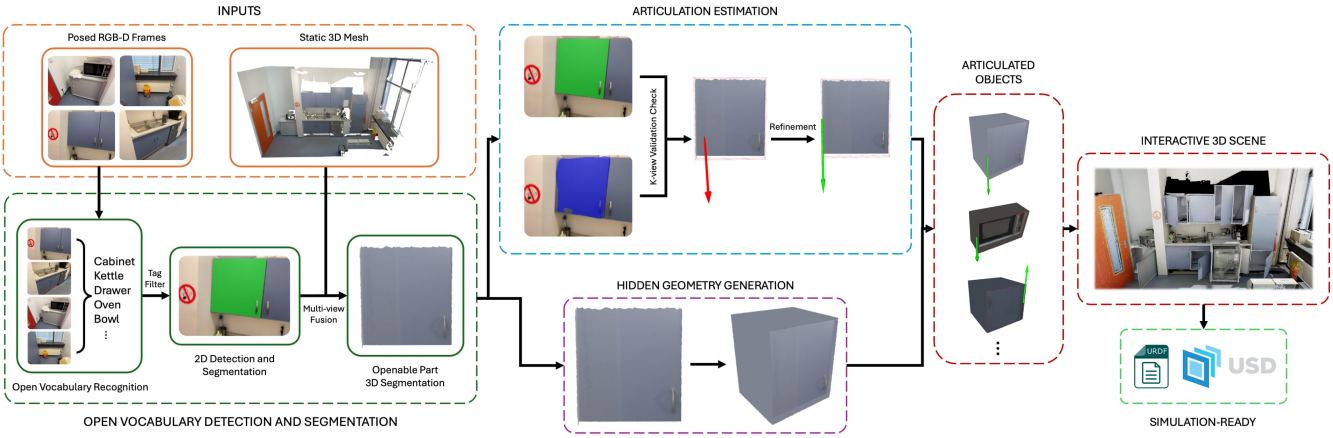


Fig. 2. Overview of REACT3D. Given a static 3D scene, our method first applies open-vocabulary detection to identify openable objects and segmentation to extract their movable parts. We then estimate articulations and generate hidden geometry to obtain interactive objects. Finally, they are integrated with the static background to produce a simulation-ready interactive scene.

`.glb`) to simulation-ready formats such as URDF and USD. These converted scenes support physical simulation and enable robotic agents to navigate and interact with articulated objects. Our method provides a reliable and efficient solution for obtaining large-scale, physics-aware 3D assets, without requiring additional real-world data collection or computationally intensive generation. By combining the abundance of static 3D datasets with the strengths of vision-based perception models, our approach offers a scalable and generalizable pathway toward dynamic scene generation. The output of our pipeline can be seamlessly integrated into a wide range of renderers, simulators, and ecosystems, such as ROS, Isaac Sim, PyBullet, and Open3D, enabling flexible deployment across diverse downstream tasks. In summary, our key contributions are:

- We present REACT3D, an efficient, automated workflow that leverages vision foundation models and vision-language models (VLMs) to recover object articulations from static 3D scenes and generate physically-enabled 3D scenes.
- We provide a comprehensive evaluation of our pipeline against baseline methods.
- We demonstrate the utility of our output assets across various platforms, including renderers, physics simulators, and the ROS ecosystem.

II. Related Work

A. Articulation estimation

Articulated object motion estimation is a long-standing problem in computer vision and robotics [5]–[7]. Numerous works have explored predicting openable object parts and their motion parameters [6], [8], [9]. OPD [6] addresses this task by extending Mask R-CNN [10] to detect articulated parts and estimate their motions from single-view object images. OPDMulti [8] further generalizes OPD to handle real-world scenes containing multiple objects. In addition, 3DOI [9] proposes a transformer-based model to estimate the physical properties and affordance of objects using a single image and 2D query points as input.

Another line of work [7], [11], [12] focuses on predicting articulation directly on scene-level point clouds. SceneFun3D [11] introduces a large-scale dataset aimed at understanding object functionalities and affordances in real-world 3D environments, along with baseline models for predicting motion parameters associated with interactions involving functional elements (e.g., opening a drawer by pulling its handle). Articulate3D [12] builds upon ScanNet++ [2] annotated scenes to provide object-part hierarchy annotations and articulation parameters. However, scene-level point clouds often lack the fine-grained details of 2D images, and their annotations remain sparse. In this work, we propose an open-vocabulary approach to segment and articulate openable parts by leveraging off-the-shelf image-based articulation estimators together with segmentation foundation models.

B. Articulated scenes from static observations

Early efforts on building interactive 3D scenes often required observing objects in multiple states or under user interaction. For instance, Ditto [13] reconstructs part geometry and joints by comparing 3D observations before and after human interaction. Similarly, PARIS [14] uses two sets of multi-view images captured under different articulation states to jointly recover part-level shape and motion parameters. Weng *et al.* [15] likewise leverage two RGB-D scans of an object at different articulations to learn a neural implicit twin with accurate kinematics. While these methods achieve faithful motion recovery, their reliance on paired observations or physical trials limits scalability. Some works have explored video-based static capture: e.g., ArtGS [16] employs 3D Gaussian splatting on casually captured video frames to reconstruct photorealistic objects with movable parts, but still benefits from careful segmentation or coarse templates to handle occluded regions. Recently, FunRec [17] and iTACO [18] further explore reconstructing articulated scenes directly from interaction videos. Overall, approaches that demand explicit state changes incur significant manual interaction overhead and scale poorly to large-scale scene conversion.

Recent research instead focuses on recovering articulated scenes directly from single-state observations using

learned priors and foundation models. At the object level, Real2Code [19] proposes to generate programmatic code describing parts and joints given RGB images, using VLM to enforce structural consistency. Articulate-Anything [20] retrieves a similar 3D model from a shape library and infers plausible joints, enabling articulation for arbitrary objects but at the cost of fidelity to the actual geometry. Other methods tackle full scenes from minimal input: Digital Cousins [21] and URDFormer [22] generate an interactive scene from a single RGB image by leveraging prior knowledge. However, single-image approaches often yield inconsistent or non-faithful results, since a single view provides only partial evidence. More comprehensive pipelines integrate multi-view reconstruction and segmentation cues. DRAWER [4] reconstructs scenes from posed RGB frames using foundation models for part segmentation and articulation estimation, together with neural fields for appearance, achieving state-of-the-art realism and accurate articulations. These advances demonstrate a clear trend toward zero-shot articulated scene generation from static inputs, but challenges remain in balancing fidelity, physical accuracy, and scalability.

III. Method

A. Preliminaries

Problem formulation. Given a static 3D scene representation $\mathbf{S}$, our goal is to construct an interactive digital twin $\hat{\mathbf{S}}$ that augments the original geometry with articulated objects while preserving the static background (Fig. 2). The input $\mathbf{S}$ consists of a point cloud or a mesh $\mathbf{P}$ with per-vertex colors, captured by a registered RGB-D sensor array, together with a set of calibrated frames $\{(\mathbf{I}_i, \mathbf{D}_i, \mathbf{T}_i, \mathbf{K}_i)\}_{i=1}^{N}$, where $\mathbf{I}_i$ is an RGB image, $\mathbf{D}_i$ its depth map, $\mathbf{T}_i \in SE(3)$ the extrinsic pose, and $\mathbf{K}_i \in \mathbb{R}^{3\times 3}$ the intrinsic matrix. The output $\hat{\mathbf{S}}$ is decomposed as

$$\hat{\mathbf{S}} = f(\mathbf{S}) \;=\; \big(\mathbf{P}_{bg},\ \{\mathbf{O}_j\}_{j=1}^{M}\big), \quad \mathbf{O}_j = \big(\mathbf{p}_j,\ \phi_j,\ \mathbf{b}_j\big),$$

where $\mathbf{P}_{bg}$ denotes the static background and each interactive object $\mathbf{O}_j$ is defined by its movable part $\mathbf{p}_j$, articulation parameters ϕ_j, and completed hidden geometry $\mathbf{b}_j$. The articulation is parameterized as

$$\phi_j = (t_j, \mathbf{o}_j, \mathbf{a}_j, \rho_j),$$

with joint type $t_j \in \{\text{prismatic, revolute}\}$, origin $\mathbf{o}_j \in \mathbb{R}^3$, axis $\mathbf{a}_j \in \mathbb{R}^3$, and motion limits ρ_j.

Openable object. We define an openable object as one that contains a movable part actuated by either a prismatic or a revolute joint [23]. For a prismatic joint, the part translates along a fixed axis $\mathbf{a}$ with displacement $s \in [0, \rho]$, while for a revolute joint, it rotates about a unit axis $\mathbf{a}$ through origin $\mathbf{o}$ with angle $\theta \in [0, \rho]$. The motion of any point $\mathbf{x}$ belonging to the movable part in its closed configuration is

$$\mathbf{x}' \;=\; \begin{cases} \mathbf{x} + s\,\mathbf{a}, & \text{prismatic,} \\ R(\mathbf{a}, \theta)(\mathbf{x} - \mathbf{o}) + \mathbf{o}, & \text{revolute,} \end{cases}$$

where $R(\mathbf{a}, \theta) = \exp([\mathbf{a}]_\times \theta)$ denotes the axis-angle rotation and $[\mathbf{a}]_\times$ is the skew-symmetric matrix of $\mathbf{a}$. This yields a compact representation of each movable part's motion, parameterized by $(t, \mathbf{o}, \mathbf{a}, \rho)$, which serves as the basis for kinematic prediction.

B. Openable object detection and segmentation

Interactive scene understanding requires semantic knowledge of object affordances beyond static 3D geometry. The detection module is utilized to identify candidate interactive instances, while part segmentation delineates movable components from the structure to obtain aligned meshes. This decomposes the static scene into instance-level proposals for downstream articulation estimation, determining the extent of interactivity in the resulting scene.

Open-vocabulary 2D detection and segmentation. Indoor scenes include diverse openable objects not captured by standard labels, motivating an open-vocabulary detection strategy to maximize interactive coverage. We therefore first process the RGB frames $\{\mathbf{I}_i\}_{i=1}^{N}$ of the static scene to recognize semantic tags of objects present in the scene using RAM++ [24]. Let $\mathcal{T} = \bigcup_{i=1}^{N} \texttt{RAM++}(\mathbf{I}_i)$ denote the aggregated tag set. We then apply a VLM-based filtering method built on LLaVA [25] to retain only tags that satisfy our definition of openable objects, yielding $\mathcal{T}_{\text{open}} = \{\, t \in \mathcal{T} \mid f_{\text{VLM}}(t) = 1 \,\}$. Finally, Grounded SAM [26] is applied with prompts $\mathcal{T}_{\text{open}}$ on each frame $\mathbf{I}_i$ to localize and segment movable parts, producing a set of 2D instance masks $\{\mathbf{m}_{i,k}\}_{k=1}^{n_i}$, where n_i denotes the number of detected movable parts in frame $\mathbf{I}_i$, which are post-processed with hole filling to close interior gaps, as Grounded SAM can miss door handles. Unlike prior approaches restricted to predefined labels, our open-vocabulary method mitigates label bias and yields better coverage of long-tail openable objects.

2D-to-3D segmentation via multi-view fusion. Zero-shot 3D instance segmentation methods [27] operating on point clouds or meshes rarely yield reliable *part*-level segmentation. We therefore adopt a 2D-to-3D multi-view fusion paradigm, similar to DRAWER, to lift robust 2D masks into 3D. For each frame, we construct the model-view-projection matrix from $\mathbf{T}_i, \mathbf{K}_i$ to recover the camera view. The scene mesh is then rasterized under each view, and the front-most faces covered by each mask are selected as its 3D projection. To recover the full structure of each object, as they are not always fully visible from a single view, we fuse the per-mask projections by building a face graph and applying Louvain community detection [28]. Unlike DRAWER, which retains only one mask above the Intersection over Union (IoU) threshold for each fusion result, our approach ranks the masks by IoU and preserves the top-k views that exceed the threshold, providing multiple perspectives that improve the robustness of subsequent articulation estimation. In addition, we replace its VLM-based filtering of part masks, which discards candidates without visible handles, with an alternative validation step at the articulation estimation stage (Sec. III-C). This design allows the detection of openable objects even when handles are absent or visually nonsalient, as shown by the comparison in Fig. 3. We then re-segment each object by invoking SAM with point prompts seeded from previously fused 3D proposals.

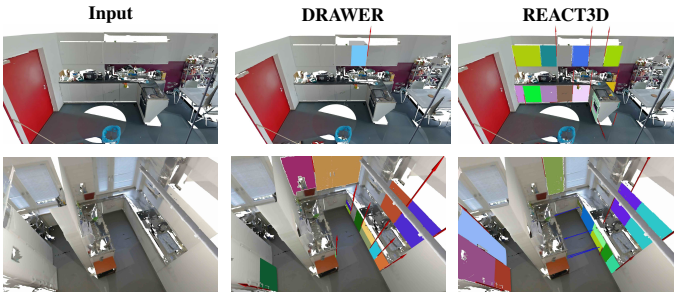


Fig. 3. Qualitative comparison. Red and blue arrows denote revolute and prismatic joints. Our method outperforms DRAWER on handle-less objects (top) and correctly classifies prismatic drawers that DRAWER mistakes for revolute joints (bottom).

Another challenge in 2D-to-3D lifting is that segmentation often produces the whole object rather than its movable part. To address this, we refine each segmented 3D mesh by applying RANSAC [29] to estimate an approximately vertical, finite-thickness plane and retain the estimate with the largest contour area. Naive planar clipping, however, may fail to capture protruding handles. To preserve such structures while discarding static body regions, let $\mathbf{n}$ denote the normal of the estimated plane and $\mathbf{v}_i$ the viewing direction derived from the camera pose $\mathbf{T}_i$ of the frame that produced the segmentation. We choose the inward-pointing normal such that $\langle \mathbf{n}, \mathbf{v}_i \rangle > 0$, and discard the geometry behind the plane along this direction. This produces high-quality 3D movable-part meshes $M_j^{\text{obj}} = (V_j, F_j)$, where V_j and F_j denote the vertex and face sets of the segmented mesh, together with the top-k RGB views and their associated masks $\{\mathbf{m}_{j,i}\}_{i=1}^{k}$. The resulting part meshes are spatially aligned with the input static scene and provide the foundation for generating interactive objects.

C. Articulation estimation

To enable meaningful manipulation, each movable part must be endowed with kinematic parameters that define its motion. Accordingly, for every movable part, we estimate an articulation tuple $\phi(m) = \big(t, \mathbf{o}, \mathbf{a}, \rho\big)$. In our framework, we leverage OPDMulti [6], [8] to obtain initial articulation estimates and corresponding movable part masks.

Candidate filtering. Since OPDMulti can detect multiple movable parts from a single RGB frame, it offers the opportunity to further validate the candidate openable objects obtained from the segmentation stage. For each segmented 3D mesh, we utilize the pre-saved top-k best-view images and their SAM-generated masks $M_{\text{SAM}} = \{\mathbf{m}_i\}_{i=1}^{k}$. We then run OPDMulti on these views and compare its predicted movable-part masks M_{OPDM} against M_{SAM}. A candidate is considered valid if the IoU between the two masks exceeds a threshold in at least one of the top-k views; otherwise, the 3D mesh is discarded as an invalid case.

Articulation refinement. While OPDMulti provides a reasonable initial articulation, direct application to interactive objects may fail to reproduce realistic joint behavior. To recover faithful interactions, we refine the initial estimates for each validated movable part by computing an oriented bounding box (OBB) of the part mesh with COMPAS [30]. The OBB is denoted $(\mathbf{U}, \mathbf{c}, \mathbf{s})$, where $\mathbf{U} = [\mathbf{u}_1\, \mathbf{u}_2\, \mathbf{u}_3]$ is a set of orthogonal edges, $\mathbf{c}$ is the center, and $\mathbf{s} = (s_1, s_2, s_3)$ are the side lengths with $s_1 \geq s_2 \geq s_3$. Since openable parts are typically plate-like, we define the front face P_{front} as the OBB face spanned by the two longest edges $\mathbf{u}_1$ and $\mathbf{u}_2$ with unit normal $\mathbf{n}_{\text{front}} = \mathbf{u}_1 \times \mathbf{u}_2$. Let the initial articulation be $\phi(m) = (t, \mathbf{o}, \mathbf{a}, \rho)$. For prismatic joints, we update only the joint axis direction by aligning it with the front-face normal,

$$\mathbf{a}' \;=\; \text{sign}\big(\langle \mathbf{a}, \mathbf{n}_{\text{front}} \rangle\big)\, \mathbf{n}_{\text{front}}, \qquad \phi'(m) = (t, \mathbf{o}, \mathbf{a}', \rho),$$

which preserves the origin $\mathbf{o}$ while enforcing translation orthogonal to P_{front}. For revolute joints, we first select the in-plane principal direction most consistent with the initial axis,

$$\mathbf{l}_s \;=\; \arg\max_{\mathbf{v} \in \{\mathbf{u}_1, \mathbf{u}_2\}} \big|\langle \mathbf{a}, \mathbf{v} \rangle\big|, \qquad \mathbf{a}' \;=\; \text{sign}\big(\langle \mathbf{a}, \mathbf{l}_s \rangle\big)\, \mathbf{l}_s.$$

Next, we place the origin by comparing the joint axis line defined by the initial axis $L(\mathbf{o}, \mathbf{a}) = \{\mathbf{o} + s\,\mathbf{a} \mid s \in \mathbb{R}\}$ to the two OBB front-face edges that are parallel to $\mathbf{l}_s$. We compute the shortest distance from L to each of these edges, select the closer one, and take its midpoint. This point is then projected along $\mathbf{n}_{\text{front}}$ onto the mid-surface equidistant between P_{front} and its parallel opposite OBB face P_{back}, where the projected point serves as the refined joint origin $\mathbf{o}'$. This procedure anchors the origin to the mid-surface edge most consistent with the initial axis while ensuring the most appropriate location within the part's thickness. Finally, we update

$$\phi'(m) = (t, \mathbf{o}', \mathbf{a}', \rho).$$

Intuitively, the prismatic axis is made orthogonal to the part face, whereas the revolute axis is aligned with the dominant in-plane edge and centered within the part thickness; in both cases the axis sign is chosen to minimize angular deviation from the initialization. This refinement aligns the estimated kinematic parameters with the observed geometry of the part, yielding more realistic motion.

D. Hidden geometry generation

The input static 3D scene encodes only the visible surface geometry of objects. Consequently, even though articulation estimation enables object interaction, the interior behind the movable part remains unmodeled upon opening. This creates visually unrealistic artifacts and, more critically, limits practical downstream applications such as robotics, where the interior volume is essential for manipulation. To construct a complete interactive object, we generate the hidden geometry through cavity completion, producing consistent box-like structures that approximate the container behind each movable part. Fig. 4 demonstrates the process of interactive object generation.

As in Sec. III-C, we compute an OBB for each movable part and take the face formed by its two longest edges as the front face P_{front}. These edges define the width and height of the inner box, while its depth is measured along the inward-pointing normal of the front face, $\mathbf{n}_{\text{front}} = \mathbf{u}_1 \times \mathbf{u}_2$. We define the inner-box depth as

$$d_{\text{in}} \;=\; \min\big(\, d_{\text{image}},\; d_{\text{hit}},\; d_{\text{mesh}} \,\big),$$

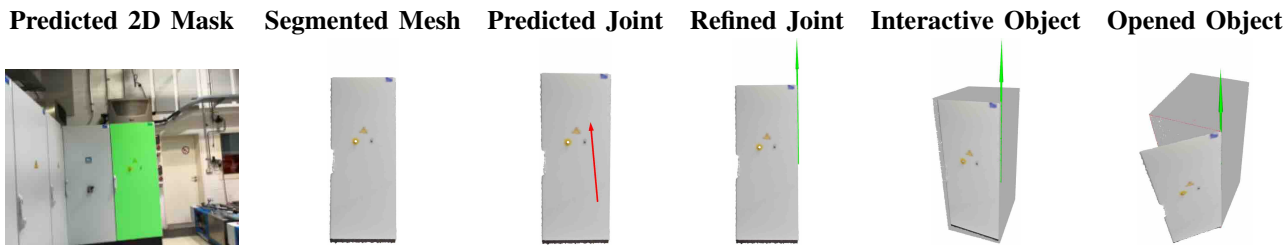


Fig. 4. Pipeline for interactive object generation. From left to right, the figure shows key intermediate results of interactive object generation. In the last column, the thin red line highlights the contour of the base part.

and then instantiate a rectangular cavity whose front coincides with P_{front} and whose extent is d_{in} along $+\mathbf{n}_{\text{front}}$. Here, d_{image} is an image-derived bound given by the farthest background depth in the RGB-D frame used to segment the part mesh, projected onto $+\mathbf{n}_{\text{front}}$. d_{hit} is obtained by probing along $+\mathbf{n}_{\text{front}}$ from the OBB centroid to the nearest scene intersection, and taking its depth if a supporting plane can be fitted there using RANSAC. Finally, d_{mesh} is the boundary depth of the static scene mesh, serving as an upper bound to prevent leakage outside the scene.

E. Interactive Scene Integration

Our goal is to produce a complete interactive scene that can be directly used in downstream applications. The output includes interactive objects consisting of movable parts with refined articulation and completed internal geometry, all without duplicates, together with the textured static background integrated into a unified representation.

Duplicate removal. To ensure clean assembly of the final interactive scene, we perform a last deduplication pass. Let $\{S_i\}$ denote movable-part meshes with point sets $\{X_i\}$. We compute the IoU directly on these point sets, $\text{IoU}(X_i, X_j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}$. The procedure has two stages. *Stage 1 (pairwise pruning):* for every pair (i, j), if $\text{IoU}(X_i, X_j) \geq \tau_{\text{dup}}$, we keep the mesh with greater information content (measured by point count) and discard the other, producing a reduced set $\{\hat{S}_i\}$ with point sets $\{\hat{X}_i\}$. *Stage 2 (subdivision pruning):* duplicates can also appear when a larger mesh can be represented by the union of several smaller, complementary meshes while no single pair exceeds τ_{dup}. We sort $\{\hat{S}_i\}$ by $|\hat{X}_i|$ in descending order and iterate from large to small so that larger parts can be explained by smaller ones first. For each $\hat{S}_i$, we collect candidate smaller meshes $\mathcal{C}_i = \{\, j : |\hat{X}_j| < |\hat{X}_i|,\ \text{IoU}(\hat{X}_i, \hat{X}_j) \geq \tau_{\text{low}} \,\}$, where τ_{low} is a small IoU threshold, and enumerate their combinations. If there exists a subset $\mathcal{U} \subseteq \mathcal{C}_i$ such that $\text{IoU}\Big(\hat{X}_i,\ \bigcup_{j \in \mathcal{U}} \hat{X}_j\Big) \geq \tau_{\text{dup}}$, we remove $\hat{S}_i$ as a redundant superset. In the end, we remove all redundant movable parts together with their associated inner boxes before assembling the final interactive scene.

Scene assembly. We integrate interactive objects into the original static scene by carving the background geometry using the world-frame OBBs of deduplicated movable parts. Let the original static scene mesh be $\mathbf{P}$ and the set of movable-part OBBs be $\{B_j\}$. To remove residual fragments from imperfect segmentation and spurious points in the original scan, we delete all vertices lying inside any B_j together with their incident faces. This yields a cleaned background mesh $\mathbf{P}_{\text{rem}} = \mathbf{P} \setminus \bigcup_j B_j$. We then assemble the final interactive scene by combining $\mathbf{P}_{\text{rem}}$ with the interactive objects, ensuring a clean integration without interpenetration.

Fig. 5. Qualitative results of REACT3D. Static input scenes from ScanNet++ and the interactive outputs generated by REACT3D, visualized in Isaac Sim.

Texture generation. As both the original scene mesh and the segmented movable-part meshes provide only per-vertex colors, we generate texture maps based on InstantTexture [31]. We first perform UV unwrapping with Xatlas [32] on the movable parts and the cleaned remainder mesh. Given the UV parameterization, we rasterize each triangle in atlas space and compute per-texel colors by barycentric interpolation of the source vertex colors, yielding dense textures aligned to the unwrapped charts. We then repair uncovered texels caused by sampling gaps or occlusions via inpainting, followed by smoothing with low-pass filtering to attenuate aliasing across chart boundaries. Finally, the generated textures are bound to their corresponding meshes and, together with the refined articulation parameters, exported as `.dae` assets that are referenced in the scene description files.

Export and simulator support. To ensure portability across visualization and simulation stacks, we export each interactive scene in two widely adopted formats: *URDF* for the robotics ecosystem and *USD* for graphics/physics engines. The exports package the cleaned background, articulated interactive objects, and textures, enabling direct deployment in PyBullet [33], ROS [34], and Isaac Sim [35]. Fig. 5 shows the simulated interactive scenes generated by REACT3D from different inputs in Isaac Sim. For ROS, we additionally provide a lightweight GUI for manipulating the interactive scene during visualization and benchmarking (see Fig. 6).

IV. Experiments

A. Experimental Setup

Datasets. To comprehensively evaluate our method and baselines, we use ScanNet++ [2], a widely adopted dataset that

TABLE I

QUANTITATIVE RESULTS ON OPENABLE OBJECT DETECTION. OPENABLE OBJECT DETECTION ON 30 SCANNET++ SCENES RICH IN OPENABLE OBJECTS. WE REPORT PRECISION, RECALL, AND F1 AT IOU THRESHOLDS $\tau \in \{0.25, 0.50\}$. *FOR URDFORMER, τ DENOTES THE COVERAGE RATIO, WHILE FOR OTHER METHODS IT DENOTES IOU. †FOR URDFORMER, THE NUMBER REFLECTS ONLY OPENABLE OBJECTS VISIBLE IN THE SELECTED REGION WITH THE HIGHEST DENSITY OF OPENABLE OBJECTS, SINCE THE METHOD TAKES A SINGLE RGB IMAGE AS INPUT.

Method	**#Openable Objects**	$\tau^* = 0.25$			$\tau^* = 0.50$		
		Precision ↑	**Recall ↑**	**F1 Score ↑**	**Precision ↑**	**Recall ↑**	**F1 Score ↑**
URDFormer	226†	0.429	0.226	0.296	0.361	0.190	0.249
DRAWER (GT-mesh)	540	0.434	0.141	0.213	0.326	0.106	0.159
REACT3D	540	**0.731**	**0.393**	**0.511**	**0.666**	**0.357**	**0.465**

The best result for each metric is highlighted in **bold**.

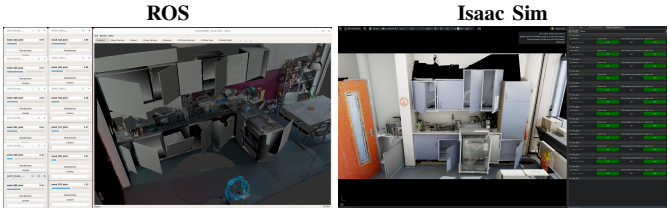


Fig. 6. Manipulation GUIs. Interfaces in ROS and Isaac Sim enabling per-object articulation control and benchmarking.

provides high-quality posed RGB-D reconstructions across diverse indoor environments. We select 30 scenes rich in articulated objects, ranging from single rooms to full apartments. For quantitative evaluation, we adopt Articulate3D [12] as ground truth, which augments ScanNet++ with part-level semantic labels and articulation annotations (joint type, origin, axis, and motion range). Our evaluation focuses on objects with open-close motion, while small controls such as buttons and knobs are excluded. For benchmarking, we evaluate our method against the baselines URDFormer and DRAWER.

Evaluation metrics. We assess performance along two dimensions: movable-part detection and articulation parameters. For detection, we compute vertex-level IoU over all pairs of predicted movable-part meshes and ground-truth part meshes from Articulate3D in world coordinates. As the predicted parts are segmented directly from the input static scene, both predictions and ground truth reside in the same coordinate frame, requiring no additional alignment. A prediction is counted as a true positive if its IoU with at least one ground-truth part exceeds τ. We report precision, recall, and F1 at $\tau \in \{0.25, 0.50\}$ across all scene objects.

For articulation, we consider only true positive detections, evaluating each predicted joint against its matched ground-truth counterpart. We first assess the predicted joint type and report joint-type accuracy. Following the MultiScan protocol [7], articulation is evaluated for parts with a correct type match using two metrics: (i) *Minimum Distance (MD)*, the shortest distance between the predicted and ground-truth joint lines, which better captures positional error than origin-to-origin distance since origin differences along the joint axis do not alter the induced motion; and (ii) *Orientation Error (OE)*, the angle between the predicted and ground-truth joint axes, capturing directional error. As prismatic motion is origin-invariant, we report only OE for prismatic joints, while both MD and OE are reported for revolute joints.

To ensure fair comparison across methods with different articulation priors, we adopt a three-way joint-type taxonomy: prismatic, horizontal revolute, and vertical revolute. Most indoor revolute mechanisms are near-vertical, so methods that always predict vertical axes gain an artificial advantage if the revolute joint is treated as a single class in evaluation, yielding deceptively low OE. Conversely, methods that explicitly model horizontal axes can be penalized with errors near $90°$. To mitigate this bias, we classify a revolute joint as horizontal if its axis forms less than $45°$ with the ground plane, and vertical otherwise. Joint-type accuracy and OE are reported under this taxonomy, revealing true motion behavior while isolating differences in method capability.

Implementation Details. The threshold values utilized in our pipeline were determined empirically and stay static across all experiments. We conducted a sensitivity analysis on the parameters of the non-learning algorithms, which demonstrates the robustness of our parameter choices.

B. Baselines

URDFormer [22] generates an interactive scene from a single RGB image without guaranteed metric scale or an explicit mapping to the source 3D scene. For fair comparison, we adopt a concise placement-scaling protocol. For each static scene, we select a region rich in potentially interactive objects and capture a posed RGB-D image. We run the official model, extract per-object 3D bounding boxes and a global box, estimate a global scale s as the ratio between the vertical extent of the ground-truth region and that of the predicted global box, then scale and place the prediction back into the static scene using the pose and depth. As the output is not segmented from the scene mesh, vertex-level IoU is inapplicable. We therefore evaluate detection via 3D coverage: for a predicted box B and a ground-truth movable-part mesh M with vertices $V(M)$ in world coordinates, $\mathrm{cov}(B, M) = \frac{|V(M) \cap B|}{|V(M)|}$ and a part is detected if $\max_B \mathrm{cov}(B, M) \geq \tau$. Articulation is then evaluated with the same protocol and metrics defined above.

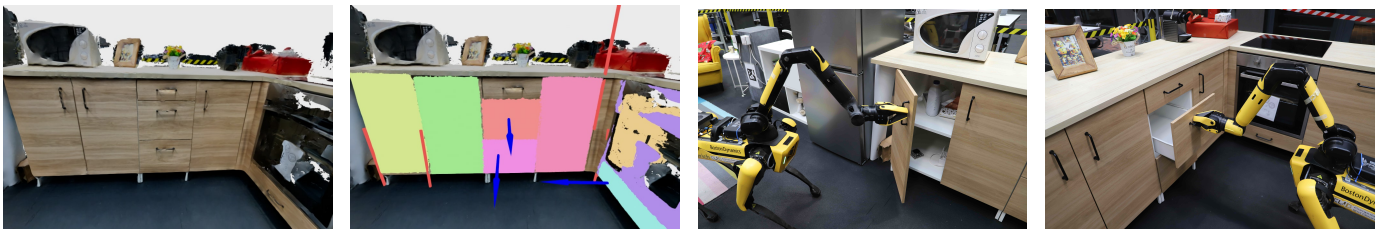
DRAWER [4] reconstructs a scene mesh from RGB frames via monocular depth and normal estimation and generates an interactive scene on top. For fair comparison on ScanNet++, we bypass reconstruction and directly use ground-truth meshes

Fig. 7. Qualitative comparison of generated interactive scenes. For each input static scene, we show results produced by each native method without any adaptation or modification. Visualizations follow each method's native support: URDFormer is rendered in PyBullet, while DRAWER and REACT3D are rendered in Isaac Sim. *Input corresponds to the ScanNet++ scene mesh, the posed RGB-D frames, or a captured image of the scene.

TABLE II
QUANTITATIVE RESULTS ON ARTICULATION ESTIMATION AND MOD EVALUATION. ARTICULATION EVALUATION ON THE SAME 30 SCENES. JOINT TYPE ACCURACY IS COMPUTED OVER CORRECTLY DETECTED MOVABLE PARTS, WHILE MINIMUM DISTANCE (MD) AND ORIENTATION ERROR (OE) ARE REPORTED ONLY FOR PARTS WITH CORRECTLY PREDICTED JOINT TYPES. *† CONDITIONS ARE THE SAME AS IN TAB. I.

Dataset/Method	#Openable Objects	$\tau^* = 0.25$							$\tau^* = 0.50$						
		Articulation			MOD [%]↑				Articulation			MOD [%]↑			
		Joint Acc. [%]↑	MD [m]↓	OE [°]↓	PDet	+M	+MO	+MOD	Joint Acc. [%]↑	MD [m]↓	OE [°]↓	PDet	+M	+MO	+MOD
URDFormer	226†	51.0	0.583	**0.50**	22.6	11.5	11.5	2.2	46.5	0.491	**0.61**	19.0	8.8	8.8	3.1
DRAWER (GT-mesh)	540	73.7	0.288	1.62	14.1	10.4	10.0	5.2	78.9	0.313	1.71	10.6	8.3	8.0	3.7
REACT3D	540	**91.5**	**0.203**	1.16	**39.3**	**35.9**	**35.9**	**25.2**	**92.2**	**0.180**	1.13	**35.7**	**33.0**	**33.0**	**23.9**

The best result for each metric is highlighted in **bold**.



(a) Static Scan (b) Output (c) Revolute (d) Prismatic

Fig. 8. Real-world Experiment. Interactive scene generated from a real-world scan, enabling robotic manipulation of revolute and prismatic joints.

as input. All downstream modules follow official settings and pretrained weights, except that the original GPT-4o component is replaced with Gemini 2.5 Flash. We denote this configuration as *DRAWER (GT-mesh)*. This protocol aligns the input domain with ours, ensuring independence from reconstruction errors in evaluation.

C. Results

Openable object detection. We first assess how many openable objects each method recovers in the generated interactive scenes. Table I compares URDFormer, DRAWER (GT-mesh), and REACT3D under the experimental setup. Note that the output scenes of URDFormer are not at true scale as explained in Sec. IV-B, whereas the other methods produce scenes at true scale. Across all metrics, our framework achieves the best results and substantially outperforms both baselines, demonstrating its superior ability to consistently and accurately recover interactive objects across diverse indoor scenes.

Articulation estimation. As shown in Tab. II, our approach achieves over 90% joint-type accuracy, substantially surpassing all baselines. It also attains the lowest MD and OE among methods with learned articulation, outperforming DRAWER, the prior state of the art with true-scale results. Although URDFormer reports the lowest OE, this is due to its heuristic strategy of presetting articulation axes to vertical or horizontal relative to the ground based on joint type. Fig. 3 shows scene-level comparisons of movable-part detection and articulation results, while Fig. 7 compares interactive scenes generated by native methods.

We further report results under the MOD setting in Tab. II, following the same experimental setup. This setting provides a cumulative view of articulation parameter correctness and reveals which metric is the primary performance bottleneck. Here, **PDet** measures detection accuracy, while **+M**, **+MO**, and **+MOD** denote progressively stricter criteria: correct motion type, motion type with OE $< 10°$, and additionally MD $< 0.25\times$ the part mesh diagonal, respectively [7]. Each stage is evaluated only if all preceding conditions are satisfied.

Ablation study. To assess the contribution of our articulation refinement module, we conduct an ablation study by comparing results with and without it. As shown in Tab. III, refinement consistently improves performance across both IoU thresholds. The reduction in minimum distance demonstrates more accurate joint localization, while the dramatic decrease in orientation error underscores its importance for reliable axis direction estimation. Overall, the results demonstrate that refinement is crucial for producing precise articulation

TABLE III
ABLATION STUDY ON ARTICULATION REFINEMENT. MINIMUM DISTANCE (MD) AND ORIENTATION ERROR (OE) ARE REPORTED AT $\tau \in \{0.25, 0.50\}$. REFINEMENT CONSISTENTLY ENHANCES ACCURACY, WITH NOTABLY LARGE IMPROVEMENTS IN OE.

Method	$\tau = 0.25$		$\tau = 0.50$	
	Avg. MD [m]↓	Avg. OE [°]↓	Avg. MD [m]↓	Avg. OE [°]↓
w/o refinement	0.226	6.72	0.213	6.64
w/ refinement	**0.203**	**1.16**	**0.180**	**1.13**

The best result for each metric is highlighted in **bold**.

parameters and achieving robust interactive scene generation.

Runtime analysis. Our framework demonstrates significantly higher computational efficiency compared to the official DRAWER baseline. In a representative scene, our approach requires 33,500s to process the full RGB-D sequence, whereas DRAWER takes 43,638s, which is ∼30% longer. Notably, our current sequential implementation offers substantial speedup potential through parallelized modular processing. Further analysis reveals that runtime is highly correlated with the total frame count. Consequently, implementing keyframe extraction can significantly reduce the runtime without compromising the quality of the results. This is evidenced by our real-world application scenario (Fig. 8), where using only keyframes achieved a runtime of 3,411s, significantly lower than processing full sequences from standard datasets.

Real-world validation. We further evaluate our pipeline by converting real-world scans into interactive digital twins for robotic manipulation. As shown in Fig. 8, a mobile manipulator successfully interacts with diverse articulated objects leveraging the recovered parameters. This demonstrates that our interactive representations are sufficiently accurate and physically grounded, enabling seamless integration with downstream robotic applications.

V. CONCLUSION

We present REACT3D, a scalable zero-shot framework that transforms static 3D scenes into interactive digital twins with articulated objects. By combining open-vocabulary detection, articulation refinement, hidden-geometry completion, and seamless simulator integration, REACT3D achieves state-of-the-art performance on scene-level openable-object detection and articulation estimation. Our results demonstrate the practical utility of REACT3D in generating simulation-ready assets, providing a foundation for large-scale research in articulated scene understanding and embodied intelligence.

REFERENCES

[1] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Berges, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi, "Habitat 3.0: A co-habitat for humans, avatars, and robots," in *ICLR*, 2024.

[2] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *ICCV*, 2023, pp. 12–22.

[3] G. Zhai, E. P. Örnek, D. Z. Chen, R. Liao, Y. Di, N. Navab, F. Tombari, and B. Busam, "Echoscene: Indoor scene generation via information echo over scene graph diffusion," in *ECCV*, 2024.

[4] H. Xia, E. Su, M. Memmel, A. Jain, R. Yu, N. Mbiziwo-Tiapo, A. Farhadi, A. Gupta, S. Wang, and W.-C. Ma, "Drawer: Digital reconstruction and articulation with environment realism," in *CVPR*, 2025.

[5] A. Sharf, H. Huang, C. Liang, J. Zhang, B. Chen, and M. Gong, "Mobility-trees for indoor scenes manipulation," *Comput. Graph. Forum*, vol. 33, no. 1, p. 2–14, Feb. 2014.

[6] H. Jiang, Y. Mao, M. Savva, and A. X. Chang, "Opd: Single-view 3d openable part detection," in *ECCV*, 2022.

[7] Y. Mao, Y. Zhang, H. Jiang, A. X. Chang, and M. Savva, "Multiscan: Scalable rgbd scanning for 3d environments with articulated objects," in *NeurIPS*, 2022.

[8] X. Sun, H. Jiang, M. Savva, and A. Chang, "Opdmulti: Openable part detection for multiple objects," in *2024 International Conference on 3D Vision (3DV)*, 2024, pp. 169–178.

[9] S. Qian and D. F. Fouhey, "Understanding 3d object interaction from a single image," in *ICCV*, 2023.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017, pp. 2980–2988.

[11] A. Delitzas, A. Takmaz, F. Tombari, R. Sumner, M. Pollefeys, and F. Engelmann, "SceneFun3D: Fine-Grained Functionality and Affordance Understanding in 3D Scenes," in *CVPR*, 2024.

[12] A.-M. Halacheva, Y. Miao, J.-N. Zaech, X. Wang, L. V. Gool, and D. P. Paudel, "Holistic understanding of 3d scenes as universal scene description," in *ICCV*, 2025.

[13] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *CVPR*, 2022.

[14] J. Liu, A. Mahdavi-Amiri, and M. Savva, "PARIS: Part-level reconstruction and motion analysis for articulated objects," in *ICCV*, 2023.

[15] Y. Weng, B. Wen, J. Tremblay, V. Blukis, D. Fox, L. Guibas, and S. Birchfield, "Neural implicit representation for building digital twins of unknown articulated objects," in *CVPR*, 2024.

[16] Y. Liu, B. Jia, R. Lu, J. Ni, S.-C. Zhu, and S. Huang, "Building interactable replicas of complex articulated objects via gaussian splatting," in *ICLR*, 2025.

[17] A. Delitzas, C. Zhang, A. Gavryushin, T. Di Mario, B. Sun, R. Dabral, L. Guibas, C. Theobalt, M. Pollefeys, F. Engelmann, and D. Barath, "Reconstructing functional 3d scenes from egocentric interaction videos," in *CVPR*, 2026.

[18] W. Peng, J. Lv, C. Lu, and M. Savva, "iTACO: Interactable Digital Twins of Articulated Objects from Casually Captured RGBD Videos," in *International Conference on 3d Vision (3dV)*, 2026.

[19] Z. Mandi, Y. Weng, D. Bauer, and S. Song, "Real2code: Reconstruct articulated objects via code generation," in *ICLR*, 2025.

[20] L. Le, J. Xie, W. Liang, H.-J. Wang, Y. Yang, Y. J. Ma, K. Vedder, A. Krishna, D. Jayaraman, and E. Eaton, "Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model," in *ICLR*, 2024.

[21] T. Dai, J. Wong, Y. Jiang, C. Wang, C. Gokmen, R. Zhang, J. Wu, and L. Fei-Fei, "Automated creation of digital cousins for robust policy learning," in *Conference on Robot Learning (CoRL)*, 2024.

[22] Q. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, D. Fox, and A. Gupta, "Urdformer: A pipeline for constructing articulated simulation environments from real-world images," in *Proceedings of Robotics: Science and Systems (RSS)*, 07 2024.

[23] X. Li, H. Wang, L. Yi, L. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," *CVPR*, 2020.

[24] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, Y. Guo, and L. Zhang, "Recognize anything: A strong image tagging model," in *CVPRW*, 2024, pp. 1724–1732.

[25] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *NeurIPS*, 2023.

[26] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, "Grounded sam: Assembling open-world models for diverse visual tasks," 2024.

[27] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask Transformer for 3D Semantic Instance Segmentation," in *International Conference on Robotics and Automation (ICRA)*, 2023.

[28] D. Do and T. H. D. Phan, "An improvement on the louvain algorithm using random walks," *Journal of Combinatorial Optimization*, 2025.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, Jun. 1981.

[30] T. V. Mele *et al.*, "COMPAS: A framework for computational research in architecture and structures." 2017-2019. [Online]. Available: https://doi.org/10.5281/zenodo.2594510
[31] D. Ebert, "Instanttexture," https://github.com/dylanebert/InstantTexture, 2024, accessed: 2025-09-10.
[32] jpcy, "xatlas: Mesh parameterization / uv unwrapping library," https://github.com/jpcy/xatlas, 2025, accessed: 2025-09-10.
[33] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.
[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, 2009.
[35] NVIDIA, "Isaac Sim." [Online]. Available: https://github.com/isaac-sim/IsaacSim