Real2Code2Real: Articulated Full-Scene Reconstruction with 3D Asset Generation

Eric Liang
Stanford University
450 Jane Stanford Way
ehliang@stanford.edu

Jacob Goldberg
Stanford University
450 Jane Stanford Way
jngoldbe@stanford.edu

Abstract

We present Real2Code2Real, a near-fully automated pipeline for reconstructing articulated, simulation-ready 3D kitchen scenes from RGBD video. Our method addresses key challenges in sim-to-real transfer by producing watertight, physically accurate mesh assets aligned to real-world geometry and articulated within a structured URDF representation. Given a single RGBD scan of a cluttered scene, we segment individual objects, generate candidate meshes using a generative model (TRELLIS), and align them to their real-world positions via a two-stage matching and refinement procedure. We then synthesize a URDF scene composed of primitive assets and replace each with its warped high-fidelity counterpart, preserving articulation and collision geometry. Our system is evaluated on kitchen environments with occluded, multi-jointed furniture and outperforms prior mesh reconstruction techniques in both geometric accuracy and physical consistency. The final scenes demonstrate high-fidelity articulation and simulation readiness, supporting downstream robotic tasks with minimal human supervision.

1. Introduction

Within robotics, reinforcement learning (RL) is an important paradigm that utilizes reward signals and constant trial-and-error for robots to learn to perform complex, multi-step tasks like object manipulation and navigation with minimal human supervision [1, 9]. To reduce real-world impracticalities like resetting scenes, many robotic RL agents are trained in simulation with programmable scenes and rewards to be transferred to similar, real-world environments [20, 16].

However, simulation-based RL often faces the sim-to-real gap, where RL agents fail to transfer their high simulation performance to the real world due to discrepancies such as inconsistent object shapes, articulation, and material properties [26, 2]. Therefore, there have been many attempts to accurately and efficiently reconstruct real-world scenes, often through scanning an environment with techniques such as NeRFs and Gaussian splatting. However, these purely image-based reconstruction methods require high-resolution videos and manual labeling, and often lead to noisy meshes that are physically inconsistent within simulations.

To address the need of accurate, articulated digital-twins of real-world scenes, we propose Real2Code2Real, an automated real-to-sim pipeline for efficient, high-quality simulations. We represent scenes with modular components produced by generative mesh models, grounded to reality with spatial alignment algorithms, and articulated with a systematic scene synthesizer. Whereas prior reconstruction methods would require comprehensive, unobstructed views of each component, our process scales smoothly with complex, occluded scenes such as with furniture. This method drastically reduces the human involvement of data collection, and provides a simple interface to augment scene components to further optimize robot RL.

1.1. Problem Statement

To evaluate our pipeline, we focus on the complex task of reconstructing and articulating an entire kitchen scene, which sees many robotic applications such as cooking, cleaning, and sorting objects. Using a baseline of both previous reconstruction methods and the ground truth scene, the task of articulating entire scenes emphasizes the global positioning of object components and important physical attributes like collisions and scale.

We structure the reconstruction task as follows: Using a single video of the entire kitchen scene as input, users can interact with the scene to accurately capture object appearances and joints. We also assume the video has a depth component, which is now available on many smartphones. Then, with a custom GUI, users specify the objects to be articulated and their semantic category by selecting and prompting a couple of frames from the video. The auto-

mated Real2Code2Real pipeline will then produce a set of files containing meshes, textures, and an XML file specifying object positions and joints to be used with simulation software.

2. Related Works

Various methods have been proposed over the years to reconstruct 3D objects and scenes given user visual input, representing both rendering and generative techniques.

Rendered reconstruction: Early breakthroughs posed scene reconstruction as an optimization problem to recreate views given large amounts of camera poses. Techniques like NeRFs optimized light fields, which downstream robotic applications like GARFIELD and Robot See Robot Do then segmented to articulate individual objects [13, 12, 11]. However, NeRFs are implicitly represented and must be recomputed at every new camera angle, making them infeasible for extracting a scene's objects. Furthermore, NeRF's focus on visual rather than physical consistency results in bumpy surfaces not suitable for simulation [24].

An alternative rendering technique, Gaussian splatting, optimizes a collection of colored Gaussian primitives to achieve explicit scene representation [10]. However, like NeRFs, Gaussians only optimize for input views, resulting in incomplete and noisy objects when the entire object isn't visible [8]. However, follow-ups such as 2D-GS and 4D-GS, which produce smoother and temporally-based scenes, have found lots of use in other 3D robotics tasks like navigation [7, 22, 3].

Scene reconstruction: Current real-to-sim pipelines like RialTo combine rendered object meshes with manual placement and articulation [21]. Although manual placement allows for greater specificity, it's much harder to scale for larger scenes and involves human discretion such as where to place bounding boxes for incomplete objects. Mandi et al. propose an automated pipeline with mesh completion and LLM joint articulation, yet face difficulty with reconstructing entire scenes due to low mesh resolution [25]. Thus, prior works demonstrate the necessity of high-quality meshes that are complete, even when input views are occluded.

3D Asset Generation: Rather than strictly adhering to input images, generative models predict entire mesh geometries by being trained on large datasets of existing meshes. Over time, these generative models have become much bigger, with the recent TRELLIS model being trained on over 500k meshes [23]. Although these meshes can generalize to a variety of objects, they still lack joint-articulation, and most importantly, spatial information relative to a global scene.

3. Dataset

Since prior works such as that by Mandi et al. primarily use non-depth input, as well as many more images per object, we compile a new RGBD dataset of 20 scans of 6 different kitchen scenes, for a total of 115 objects including drawers, cabinets, microwaves, and fridges. The RGBD video data was collected using the Record3D app on an iPhone 15 Pro equipped with LiDAR.

The RGB images have 1920 x 1440 resolution whereas the LiDAR-based depth images are 256 x 192 pixels, so we interpolate the depth images for this task. Although no other augmentation was done on the visual inputs, we apply several layers of preprocessing on the point clouds projected from the inputs to reduce noise from the low-resolution LiDAR. We first apply Open3D's outlier removal algorithm to remove noisy edge points [27]. Then, because some RGB pixels may correspond to NAN depth pixels, we apply a vectorized breadth-first search algorithm to match NAN-pixels with the closest existing pixel.

Along with fully labeled and segmented images, averaging 228 frames per object, we also saved the meshes, the input-mesh matched points, and the roughly aligned meshes for each object. This dataset serves as both a benchmark for future full-scene mesh generation and alignment attempts, as well as a dataset for kitchen-based mesh tasks in the future. Samples of the dataset for two scenes are shown in Figure 1.

4. Methods

Our approach is divided into four core stages: (1) Data collection and pre-processing, where we capture RGBD videos and segment target objects using SAM-2 [17]; (2) Mesh generation and refinement, where we iteratively reconstruct object meshes using Microsoft's TRELLIS model to select the most physically accurate mesh; (3) Mesh alignment where we estimate scale and real-world location using visual comparisons between the mesh and input frames with SuperGlue [18]; and (4) Scene synthesis and joint articulation, where we systematically generate articulated scenes using NVIDIA's Scene Synthesizer [6]. Fig. 2 offers an overview of our proposed method.

4.1. Segmentation

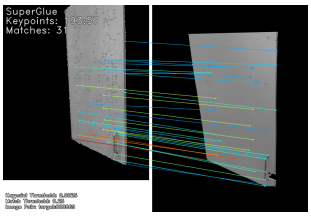
To extract relevant object components from the RGBD video stream, we combine depth information with the Segment Anything Model 2 (SAM 2) and introduce a simple GUI to efficiently prompt a couple of points for each object. The SAM 2 model was trained on a dataset of over 11-million images, making it generalizable for components of real-world scenes, and its built-in memory module requires only a single prompt per object, unlike previous works which require manual reprompting [4]. We further

(a) Packard Kitchen RGB

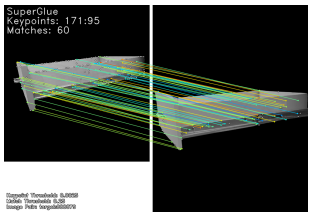
(b) Home Kitchen RGB

(c) Basement Kitchen Depth

(d) Home Kitchen Depth



(e) Packard Kitchen Matches



(f) Home Kitchen Matches

Figure 1: Custom Dataset for Kitchen Scene RGBD and Alignment

enhance accuracy by applying these prompts on the corresponding depth maps, and taking the intersection of the two masks. This helps overcome a major issue in purely visual approaches, where objects that blend in with the background produce a noisy mask that interferes with mesh generation [14].

4.2. Mesh Generation

Following segmentation, each object is passed through the TRELLIS model to generate a watertight 3D mesh from multi-view RGBD observations. The TRELLIS model encodes image features using the DINOv2 model [15], and then uses rectified flow transformers to generate a latent object representation, which is decoded into various formats like Gaussians or meshes. TRELLIS' large dataset of detailed meshes allows it to predict entire object bodies, even when occluded images are provided as input.

Due to the importance of image quality and variety, we divide each object's input images into 20 intervals, which produced the best results during qualitative hyperparameter tuning. Within each interval, we evaluate each frame by how large and centered the object is, and select the frame with the largest, uncropped mask.

To minimize the model's hallucinations, where an incomplete view of the object leads to incorrect mesh predictions, we sample TRELLIS up to 10 times for the same object, randomly varying the input images and parameters. We then compare the bounding boxes and volume of the generated meshes, selecting the mesh with measurements closest to the average of all the samples to even out any noise from the generative model.

4.3. Mesh Alignment

The generated meshes are initially in an arbitrary scale and coordinate frame, requiring alignment to ground truth scene geometry for each individual object. To address this, we render RGB and depth images from the generated mesh using known camera intrinsics.

Then for each of the input frames, we evaluate its similarity with all of the rendered angles. For each of these image pairs, we use SuperPoint to detect keypoints in both images, and SuperGlue to match these keypoints with a graph-based attention model for visual context [5]. We then select the rendered angle with the highest correspondence, and project these 2D matched points into their 3D coordinates using depth and camera intrinsic information:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \dfrac{(u - c_x)\, Z}{f_x} \\ \dfrac{(v - c_y)\, Z}{f_y} \\ Z \end{pmatrix}. \tag{1}$$

where (u, v) are the pixel coordinates, Z the depth, and camera focal lengths f_x, f_y and optical center (c_x, c_y).

Next, we estimate a rigid transformation between the mesh and ground truth using the 3D correspondences and a RANSAC-like voting algorithm. We first estimate the scale local to each input frame k by iterating through pairs of points $(\mathbf{q}_i^{(k)}, \mathbf{q}_j^{(k)})$ from the input point cloud and their corresponding points $(\mathbf{p}_i^{(k)}, \mathbf{p}_j^{(k)})$ on the rendered point cloud. We calculate the ratio $s_{i,j}^{(k)}$ between the input distance and the rendered distance and apply it to all the rendered points. Finally, we evaluate how many pairs of points achieve a difference between the input and the scaled rendered points below a small threshold, selecting the scale $s^{(k)}$ with the most "votes".

This algorithm is then applied to find the global scale, where we apply $s^{(k)}$ to all input-rendered frame pairs and select the scale s^* with the highest votes. We repeat the process to find the best rotation and translation $(R^{(k)}, t^{(k)})$ for each input frame, which then produces the best global transformation (s^*, R^*, t^*):

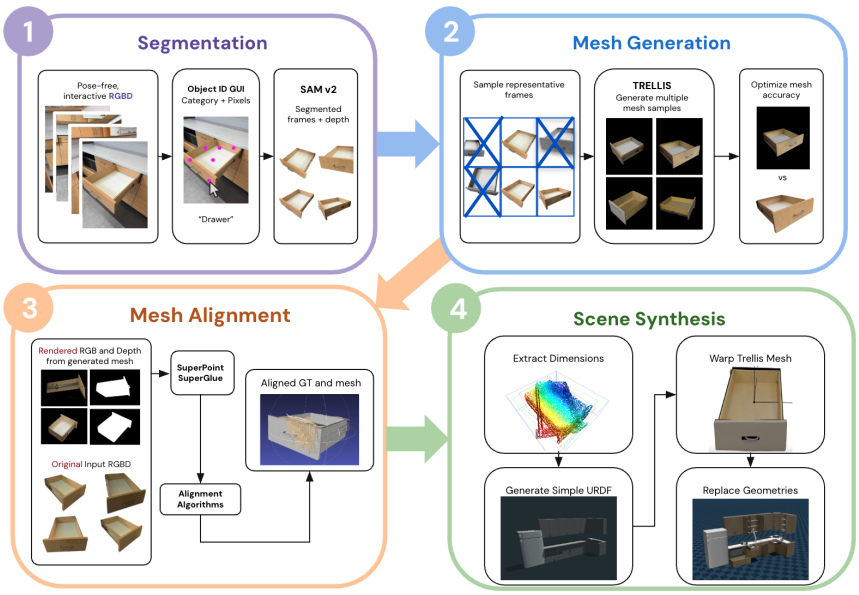


Figure 2: Overview of the proposed pipeline for any given RGBD video. 1) We segment any meaningful objects in the scene. 2) We generate meshes and algorithmically select the best one. 3) We use visual and depth information to roughly align the meshes. 4) We insert meshes into a completed, articulated scene.

$$\mu_p^{(k)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i^{(k)}, \quad \mu_q^{(k)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{q}_i^{(k)}, \tag{2}$$

$$\widetilde{\mathbf{p}}_i^{(k)} = \mathbf{p}_i^{(k)} - \mu_p^{(k)}, \quad \widetilde{\mathbf{q}}_i^{(k)} = \mathbf{q}_i^{(k)} - \mu_q^{(k)} \tag{3}$$

$$H^{(k)} = \sum_{i=1}^{N} \widetilde{\mathbf{q}}_i^{(k)}\, \widetilde{\mathbf{p}}_i^{(k)\top} = U^{(k)}\, \Sigma\, V^{(k)\top}, \tag{4}$$

$$R^{(k)} = V^{(k)}\, U^{(k)\top}, \tag{5}$$

$$t^{(k)} = \mu_q^{(k)} - s^*\, R^{(k)}\, \mu_p^{(k)}, \tag{6}$$

$$(R^*, t^*) = \arg\min_{R,t} \sum_{k=1}^{K} \left\| s^* R^{(k)} \mathbf{p}_i^{(k)} + t^{(k)} - \mathbf{q}_i^{(k)} \right\|^2 \tag{7}$$

Having iterative rounds of voting ensures all points are considered without being influenced by outliers, providing a coarse transformation which is then refined using Iterative Closest Point (ICP) to minimize residual distance between the matched points.

This two-stage alignment process produces a realistically shaped, scaled, and aligned mesh ready for substitution into the URDF scene representation (Fig. 3). It ensures consistency between the TRELLIS-generated geometry and the structured physical layout encoded in our simulator.

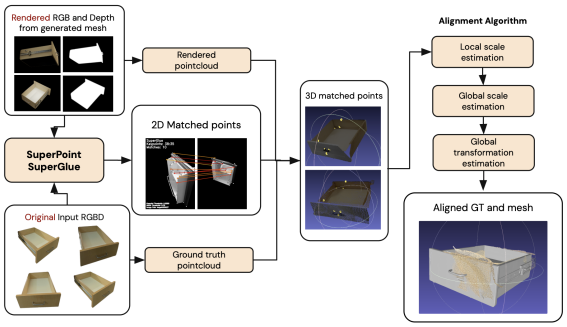


Figure 3: An overview of the alignment algorithm, where we find correspondent points between the input RGBD and the rendered mesh RGBD, project into 3D, and apply a RANSAC-like voting algorithm.

4.4. Scene Synthesis

The initial two-stage alignment process is great for creating a realistic mesh, scaled to the scene. However, the global alignment and proportion is not high enough quality for downstream applications and optimal articulation. Thus, we created a separate scene synthesis phase that takes in the aligned and scaled realistic meshes and outputs a URDF generated via NVIDIA's Scene Synthesizer package with its simple geometries with our realistic meshes. Our scene synthesis pipeline comprises two stages: (1) extracting ac-

curate box dimensions from unaligned meshes to generate a URDF of simple box primitives, and (2) aligning and warping each high-fidelity mesh so its front face exactly replaces the corresponding box primitive in the URDF.

4.5. Dimension Extraction

Raw TRELLIS meshes $\mathbf{M}$ are arbitrarily oriented, so their extents cannot be trusted directly. We therefore rotate $\mathbf{M}$ to minimize its AABB volume. Parametrizing a rotation $R(\boldsymbol{\theta})$ by three Euler angles $\boldsymbol{\theta} = (\alpha, \beta, \gamma)$, we solve

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in[0,2\pi)^3} \text{Volume}\big(\text{AABB}\big(R(\boldsymbol{\theta})\,\mathbf{M}\big)\big) \tag{8}$$

$$\mathbf{M}^* = R(\boldsymbol{\theta}^*)\,\mathbf{M}. \tag{9}$$

In practice, we initialize $\boldsymbol{\theta}$ from a few canonical orientations and run the Powell optimizer to find $\boldsymbol{\theta}^*$. Once aligned, the three extents yield width, depth, and height.

$$\mathbf{d} = \text{extents}(\text{AABB}(\mathbf{M}^*)) \tag{10}$$

We then use these dimensions as inputs to NVIDIA's Scene Synthesizer assets. We modified NVIDIA's Scene Synthesizer package to represent the most fundamental assets required in kitchen-scene generation. For example, we represent individual drawers and cabinets as separate assets rather than using the included complex kitchen assets, which bundle multiple primitives. The package generates a structured URDF of the kitchen scene composed solely of box primitives with correct joint articulation, joint limits, and collision properties, using the measurements from the dimension-extraction process.

4.6. Align, Warp, and Replace

After synthesizing a URDF asset $\mathbf{B}$ for each object, we deform the corresponding high-fidelity mesh $\mathbf{M}$ so that its "front face" coincides exactly with $\mathbf{B}$'s front face, preserving depth and maintaining all collision/articulation geometry. This involves three steps: axis alignment (including reflections), corner extraction, and local-frame affine warping (Fig. 4).

4.7. Axis Alignment Including Reflections

First, we center $\mathbf{M}$ and $\mathbf{B}$ and compute $\mathbf{d}_{\text{mesh}}$, $\mathbf{d}_{\text{URDF}}$. We then search for a signed permutation

$$R \in \{\pm 1\}^{3\times 3}, \qquad \det R = +1 \tag{11}$$

that best aligns two components of $\mathbf{d}_{\text{mesh}}$ with two components of $\mathbf{d}_{\text{URDF}}$, discarding the largest mismatch. The minimizing permutation π^* yields a permutation matrix P. Next, consider all sign-flip matrices

$$S \;=\; \text{diag}(\pm 1, \pm 1, \pm 1), \qquad \det\big(S\,P\big) = +1. \tag{12}$$

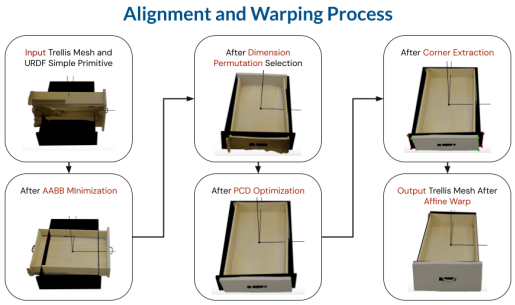


Figure 4: An overview of the three step alignment and warping algorithm, where we warp the Trellis mesh to the URDF simple primitive.

For each candidate $R = S\,P$, apply R to $\mathbf{M}_0$ (producing $\mathbf{M}_R$), sample a dense point set $\{p_k\} \subset \mathbf{M}_R$, and compute

$$D(R) \;=\; \sum_k \big\| p_k - \Pi_B(p_k) \big\|_2^2, \tag{13}$$

where $\Pi_B(p)$ is the nearest-point projection onto the mesh of asset $\mathbf{B}$. The optimal

$$R^* \;=\; \arg\min_R D(R), \qquad \mathbf{M}_{\text{canon}} \;=\; R^*\,\mathbf{M}_0, \tag{14}$$

is both axis-consistent and correctly "front-facing" as the nearest-neighbor L2 distance calculations on each point in the sampled PCD determine which orientation is the most similar.

4.8. Corner Extraction

We extract four "front-face" corners on $\mathbf{M}_{\text{canon}}$ using a purely geometric, quadrant-based procedure. First, every vertex of $\mathbf{M}_{\text{canon}}$ is projected onto the XZ-plane, yielding points (x, z). Each projected point is assigned a score proportional to its radial distance $\sqrt{x^2 + z^2}$, multiplied by a factor that gives preference to vertices with lower y-coordinate (closer to the front of the mesh). We partition the XZ-plane into the four quadrants, and in each quadrant, we select the vertex with the highest score; these four selected vertices become the source corners s_0, s_1, s_2, s_3, ordered as (top-right, top-left, bottom-left, bottom-right).

For the URDF asset $\mathbf{B}$, we identify its front-face corners by inspecting all vertices of $\mathbf{B}$ and finding those closest to the four ideal extremal points on $\mathbf{B}$'s front face. For each of these four ideal positions, we choose the actual vertex t_k of $\mathbf{B}$ that minimizes Euclidean distance. These four URDF vertices become the target corners t_0, t_1, t_2, t_3, in the same quadrant order as the source corners.

4.9. Local-Frame Affine Warping

After extracting four corresponding front-face corners on the canonical mesh $\mathbf{M}_{\text{canon}}$ (denoted $\{s_k\}$) and on the

URDF asset **B** (denoted $\{t_k\}$), we build local orthonormal frames at each set of corners by taking the centroid and edge directions. In each frame, every corner s_k (resp. t_k) is projected into a 2D "UV-plane" (dropping the common depth coordinate). Because all four front-face corners lie in exactly the same plane, their depth coordinates agree, so there exists a 2×2 matrix A and translation $\mathbf{t} \in \mathbb{R}^2$ satisfying

$$\begin{pmatrix} u'_k \\ v'_k \end{pmatrix} \;=\; A \begin{pmatrix} u_k \\ v_k \end{pmatrix} \;+\; \mathbf{t}, \quad k = 0, 1, 2, 3, \tag{15}$$

where (u_k, v_k) are the source-frame UV-coordinates of s_k and (u'_k, v'_k) are the target-frame UV-coordinates of t_k. Solving this small linear system forces those four source corners to map exactly onto their targets in UV-space.

Finally, any vertex $m \in \mathbf{M}_{\text{canon}}$ is warped by first projecting it into the same source UV-frame (and recording its depth w), applying the 2D affine map

$$(u, v) \;\mapsto\; (u^*, v^*) \;=\; A\,(u, v) \;+\; \mathbf{t}, \qquad w^* = w, \tag{16}$$

and then reconstructing in the target frame along its two in-plane axes plus the preserved depth. After undoing the initial centering translation, the warped mesh replaces the primitive asset **B** in the URDF: its front-face corners and overall depth match, while all joints and collision settings remain unchanged.

5. Experiments and Results

We run our pipeline on a variety of kitchen and non-kitchen scenes to determine its effectiveness in the following scenarios:

- How physically consistent is the generated and aligned mesh compared to real-world counterparts? How do these metrics compare with baseline approaches?
- How physically consistent is the global scene, and are such scenes viable for simulation use?
- Can the Real2Code2Real pipeline generalize to a variety of real-world scenes?

5.1. Object Reconstruction

We evaluate the Real2Code2Real on a variety of everyday scenes to demonstrate its effectiveness in producing meshes with the correct dimensions and shape. We compare our default Real2Code2Real pipeline with 2D Gaussian splatting on the same dataset by running COLMAP in order to estimate camera poses from the input images, and then reducing the output noise via clustering outlier removal and black trails stemming from the segmentation borders [19].

To determine whether our generated meshes are of the correct scale, we compute the ground truth extent D_k^{GT} and the mesh extent D_k^{R} for three orthogonal axes $k \in \{x, y, z\}$, and take the average of each axes' margin-of-error e_k fraction as follows:

Method	Drawer	Water Bottle	Microwave	Refrigerator
Garfield	22.185%	6.194%	10.185%	10.723%
2D-GS	19.092%	2.975%	8.142%	9.849%
Real2Code2Real	4.192%	3.264%	5.182%	5.172%

Table 1: The average of the three dimension's margin of error for various object reconstruction methods given the same segmented RGBD input.

$$D_k^{\text{GT}} \;=\; \max_{p \in \mathcal{P}_{\text{GT}}} p_k \;-\; \min_{p \in \mathcal{P}_{\text{GT}}} p_k, \tag{17}$$

$$D_k^{\text{R}} \;=\; \max_{q \in \mathcal{P}_{\text{R}}} q_k \;-\; \min_{q \in \mathcal{P}_{\text{R}}} q_k, \tag{18}$$

$$e_k \;=\; \frac{\left| D_k^{\text{R}} - D_k^{\text{GT}} \right|}{D_k^{\text{GT}}}. \tag{19}$$

$$E_{\text{avg}} \;=\; \frac{100\%}{3}\Big(e_x + e_y + e_z\Big) \tag{20}$$

The dimension performance of the objects are displayed in Table 1.

Additionally, we evaluate the accuracy of our shapes by calculating the intersection-over-union (IOU) between the denoised points from the RGBD ground truthand the aligned mesh. The IOU captures how well the two volumes overlap, penalizing any missing sections or protrusions from our mesh, and are calculated as:

$$\text{IoU}(\mathcal{P}_{\text{GT}},\, \mathcal{M}_{\text{R}}) \;=\; \frac{\left|\, V\big(\mathcal{P}_{\text{GT}}\big) \;\cap\; V\big(\mathcal{M}_{\text{R}}\big) \right|}{\left|\, V\big(\mathcal{P}_{\text{GT}}\big) \;\cup\; V\big(\mathcal{M}_{\text{R}}\big) \right|},$$

where $V(\mathcal{P}_{\text{GT}})$ and $V(\mathcal{M}_{\text{R}})$ represent the occupied voxels between the ground truth and reconstructed mesh, respectively. These results are shown in Table 2.

Method	Drawer	Water Bottle	Microwave	Refrigerator
Garfield	31.84%	87.24%	80.77%	78.36%
2D-GS	46.18%	97.62%	88.12%	80.84%
Real2Code2Real	91.25%	93.19%	89.44%	91.02%

Table 2: The intersection-over-union of the aligned ground-truth point cloud compared to the generated mesh for various object reconstruction methods given the same segmented RGBD input.

We see that the generative technique of Real2Code2Real performs on average better for objects found in everyday scenes, especially for more complex, multi-jointed objects. As shown in Figure 5, meshes produced by rendering methods such as 2D-GS often suffer from extraneous noise around their borders due to noisy segmentation or depth,

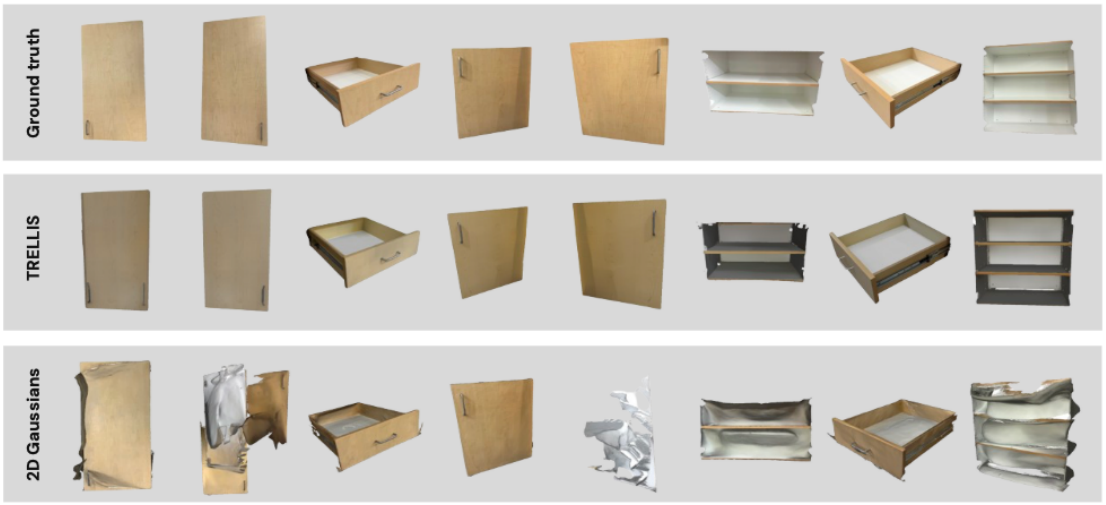


Figure 5: Comparison between input image, TRELLIS generated mesh, and 2D-GS generated mesh. We see that TRELLIS provides watertight, smooth meshes in nearly all cases.

even after removing noise as was done for these experiments. This resulted in the dimensions being skewed for rendering methods, where axes were stretched by noise to be longer than the object body; this would pose major challenges in simulation as it would result in incorrect collisions with the environment. However, Real2Code2Real aligned meshes were complete with minimal noise which ensured it's collision dimensions were accurate to the real world. However, we still see some margin of error due to TRELLIS' tendancy to hallucinate objects, which can still result in mismatched shapes and protrusions.

Additionally, we see that the Real2Code2Real aligned meshes achieved higher and more consistent IOU metrics, suggesting the aligned meshes adhered more to the real world. Unlike rendering methods, which are only able to reconstruct the visible parts of an object, resulting in occluded regions like the backs of cabinets to be malformed or hollow, Real2Code2Real's approach of selecting the best mesh of several samples ensures the mesh conforms to geometric expectations whenever input is ambiguous, as seen with the cabinet bodies in Figure 5.

5.2. Scene Reconstruction

Next, we evaluate the task of combining individual objects together to form coherent, simulation-ready scenes. This introduces new challenges such as minimizing collisions and ensuring individual components are properly aligned. We initially used 2D-GS meshes as a baseline for global scene synthesis, but the incomplete, non-manifold edges produced a static scene that was unable to move within simulation. We present the joint articulation accuracy in Table 3 evaluated the Packard Basement Kitchen (BK) and Packard Second Floor Kitchen (SFK). Cabinet doors were evaluated on the hinge joint range of motion in radians, and drawers were evaluated on the slide joint range of motion as a ratio to the drawer width. We also provide qualitative evaluations of fully reconstructed scenes in Figure 6.

Method	Cabinets BK	Cabinets SFK	Drawers BK	Drawers SFK
2D-GS	N/A	N/A	N/A	N/A
Real2Code2Real	100%	100%	95.25%	91.25%

Table 3: The joint articulation accuracy of the full-scene reconstruction compared to the ground truth for two scene reconstruction methods.

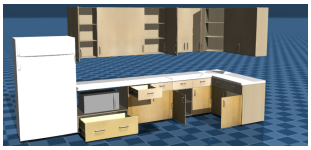
We see that Real2Code2Real is able to construct complex, articulated scenes with minimal human input. Any rotational hinges are well-approximated due to the comprehensive nature of the Scene Synthesizer package, and sliding joints such as drawers are well approximated but still slightly noisy. Furthermore, objects in the scene are smoothly aligned with one another, resulting in no joint collisions between the cabinets, drawers, and counters. This enables downstream usage, such as the ability to directly import and train robotic RL agents. However, the slight errors in dimensions may still contribute to a sim-to-real gap.

5.3. Generalized Scenes

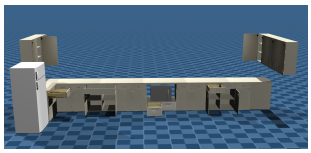
Although our dataset was focused on kitchen scenes, we test the robustness of our model by generalizing to other scenes that may be of interest to simulation RL. We present these scenes in Figure 7.

(a) BK Ground Truth

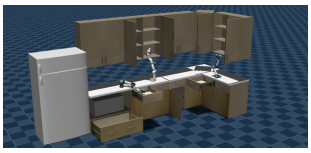
(b) SFK Ground Truth



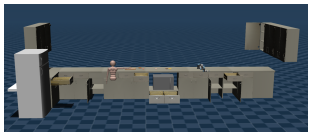
(c) BK Reconstruction



(d) SFK Reconstruction



(e) BK Downstream Usage



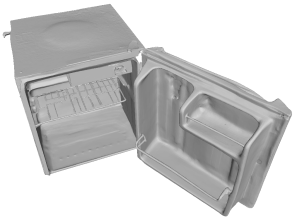
(f) SFK Downstream Usage

Figure 6: Qualitative results for global scene reconstruction on the Packard basement and second floor scenes.

(a) Meeting Ground Truth

(b) Fridge Ground Truth

(c) Meeting Reconstruction



(d) Fridge Reconstruction

Figure 7: Qualitative results for global scene reconstruction on a meeting room and mini-fridge.

We see that there is a high degree of generalization for non-kitchen scenes, even when only using the mesh alignment algorithm without the Scene Synthesizer. Kitchens often feature a higher quantity of simpler mesh geometries, whereas these generalized scenes demonstrate that Real2Code2Real also achieves high performance on smaller, more intricate scenes.

6. Conclusion and Future Works

We presented Real2Code2Real, a scalable, automated pipeline that reconstructs complex real-world kitchen environments into articulated, simulation-ready scenes. By combining generative mesh models with rigorous alignment and dimension extraction procedures, we are able to produce watertight, realistic meshes that are correctly scaled and physically consistent with the scene. Our warping and articulation methods further ensure that each object operates as expected within simulation environments via the NVIDIA Scene Synthesizer. The resulting scenes not only exhibit high visual and physical fidelity but also support robotic learning applications through accurate collision and joint modeling.

While the results are promising and the reconstructed scenes appear highly realistic, there are still limitations. Currently, the generation of the simple URDF representation—while parameterized by automatically extracted dimensions—requires manual authoring of the overall asset layout. Some spatial attributes, such as the vertical translation of wall-mounted cabinets, remain difficult to extract reliably from RGBD input and were hardcoded in our scenes. Additionally, the point cloud distance (PCD) nearest-neighbor similarity metric used in warping becomes unreliable when the generated mesh is noisy, geometrically malformed, or has hallucinations; our success is contingent on the relatively clean outputs produced by TRELLIS, which we were indeed able to obtain for our scenes.

Looking forward, we plan to fine-tune the TRELLIS model on domain-specific data such as kitchen environments to improve consistency in mesh geometry, especially under occlusion. Injecting depth features directly into the TRELLIS sparse latent transformer could further enforce geometric consistency and yield meshes that are better aligned with physical reality. This improvement would enhance downstream performance in both dimension extraction and warping. Finally, we envision a fully automated pipeline where a coding agent or script intelligently places and configures all simple URDF assets based on mesh metadata—eliminating the last remaining manual bottleneck in the real-to-sim process.

7. Contributions Acknowledgements

Eric Liang designed and implemented the mesh generation pipeline, including segmentation, multi-view sampling, TRELLIS-based mesh selection, and scale-aware alignment using SuperGlue. Jacob Goldberg designed and implemented the scene synthesis pipeline, including mesh warping, dimension extraction, and integration with the URDF-based Scene Synthesizer. Both authors contributed equally to writing the paper and evaluating results. We were advised throughout the project by our lab mentor Mandi Zhao, although we completed this project for CS231N. Our scene synthesis pipeline builds upon and significantly extends NVIDIA's open-source Scene Synthesizer package, available at: $https://github.com/NVlabs/scene_synthesizer$.

All experiments were run locally.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, Aug. 2018. arXiv:1808.00177 [cs].

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, Oct. 2018. arXiv:1810.05687 [cs].

[3] T. Chen, O. Shorinwa, J. Bruno, A. Swann, J. Yu, W. Zeng, K. Nagami, P. Dames, and M. Schwager. Splat-nav: Safe real-time robot navigation in gaussian splatting maps. *arXiv preprint arXiv:2403.02751*, 2024.

[4] Y. Chen, M.-Y. Son, C. Hua, and J.-Y. Kim. Aop-sam: Automation of prompts for efficient segmentation. *arXiv preprint arXiv:2505.11980*, 2025.

[5] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[6] C. Eppner, A. Murali, C. Garrett, R. O'Flaherty, T. Hermans, W. Yang, and D. Fox. scene_synthesizer: A Python Library for Procedural Scene Generation in Robot Manipulation. *Journal of Open Source Software*, 10(105):7561, Jan. 2025.

[7] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24*, pages 1–11, July 2024. arXiv:2403.17888 [cs].

[8] L. Huang, J. Bai, J. Guo, and Y. Guo. On the error analysis of 3d gaussian splatting and an optimal projection strategy. In *European Conference on Computer Vision (ECCV)*, pages 247–263. Springer, 2024.

[9] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, Apr. 2019.

[10] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):1–14, Aug. 2023.

[11] J. Kerr, C. M. Kim, M. Wu, B. Yi, Q. Wang, K. Goldberg, and A. Kanazawa. Robot See Robot Do: Imitating Articulated Object Manipulation with Monocular 4D Reconstruction, Sept. 2024. arXiv:2409.18121 [cs].

[12] C. M. Kim, M. Wu, J. Kerr, K. Goldberg, M. Tancik, and A. Kanazawa. GARField: Group Anything with Radiance Fields, Jan. 2024. arXiv:2401.09419 [cs].

[13] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Aug. 2020. arXiv:2003.08934 [cs].

[14] J. Montalvo, Á. García-Martín, P. Carballeira, and J. C. SanMiguel. Unsupervised class generation to expand semantic segmentation datasets. *arXiv preprint arXiv:2501.02264*, 2025.

[15] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P. Huang, S. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[16] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization, Oct. 2018. arXiv:1710.06537 [cs].

[17] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[18] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks, Mar. 2020. arXiv:1911.11763 [cs].

[19] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, Mar. 2017. arXiv:1703.06907 [cs].

[21] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation, Nov. 2024. arXiv:2403.03949 [cs].

[22] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering, Dec. 2023. arXiv:2310.08528 [cs] version: 2.

[23] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3d latents for scalable and versatile 3d generation, Dec.

2024. arXiv:2412.01506 [cs] – v2 (17 Apr 2025). Code: https://github.com/microsoft/TRELLIS.

[24] W. Xiao, R. Chierchia, R. Santa Cruz, X. Li, D. Ahmedt-Aristizabal, O. Salvado, C. Fookes, and L. Lebrat. Neural radiance fields for the real world: A survey. *arXiv preprint arXiv:2501.13104*, 2025.

[25] M. Zhao, Y. Weng, D. Bauer, and S. Song. Real2Code: Reconstruct Articulated Objects via Code Generation, June 2024. arXiv:2406.08474 [cs].

[26] W. Zhao, J. Peña Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey, Sept. 2020. arXiv:2009.13303 [cs].

[27] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.