SAM 3D: 3Dfy Anything in Images

Supplementary Material

Outline

The appendix provides additional context to the main paper; it contains additional details about the method and the implementation in SAM 3D, as well as ablations.

The structure of the appendix is as follows:

(i) **Related Work:** Extended discussion of related work.

(ii) **Data Engine details:** A more detailed description of the data collection used in the *collection step* in Section 3.2.1.

(iii) **Pretraining and Mid-Training Data:** How we collected and filtered the data used for pretraining and mid-training the Geometry and Texture & Refinement models

(iv) **Training details:** Architectural details about MoT and the VAEs. Definitions used for objectives used in each stage. Details on the Geometry and Texture & Refinement models.

(v) **Evaluations:** Introducing the details of the new **SA-3DAO** benchmark, and evaluation protocols of preference tests and quantitative metrics

(vi) **Additional experiments and qualitative examples:** Providing additional analysis and insights into the model's performance.

(vii) **Limitations:** An analysis of common failure modes, and future work

A. Related Work

3D reconstruction has been a longstanding challenge in computer vision. Classical methods leverage multiple views and include binocular stereopsis [105], structure-from-motion [36, 83, 89, 93, 94], and SLAM [10, 86]. Other strategies reconstruct by analysis (e.g., silhouettes [23]) or by synthesis via volume rendering [40], using either implicit representations [64] or explicit ones [55, 85]. Supervised deep learning methods predict voxels [98, 114], point clouds [96], or meshes [104, 106], or optimize implicit representations [56], e.g., signed distance functions (SDFs), often with high-quality output but requiring multiple views at inference. In contrast, we focus on the more restrictive setting of a single RGB image at test time.

Single-view 3D reconstruction is considerably more difficult. A large body of work trains reconstruction models with direct 3D supervision, predicting meshes [46, 117], voxels [28, 74, 107], point clouds [24, 63], or CAD-aligned geometry [30, 99]. More recently, large internet corpora of 3D object assets have enabled a line of generative approaches trained using VAE [43] latent representations [79, 112, 125]. A related challenge is recovering the full shape of occluded objects from a partial mask [111, 116]; SAM 3D handles this through data rather than architecture. However, existing generative methods are typically evaluated on simplified synthetic single-object benchmarks such as ShapeNet [11], Pix3D [88], or Objaverse [17].

Layout estimation extends single-instance estimation to full scenes. Instead of predicting a single mesh for the entire scene, SAM 3D estimates scene layout by using *mask prompts* to indicate which objects to reconstruct, and then estimates both shape and pose for each mask instance. Approaches can be decomposed into so-called model-based or model-free methods. Model-based methods decompose the problem into independent parts, first estimating shape (or pulling it from a CAD database) and then posing the objects in the scene [47, 60, 61, 82, 95, 103]. Alternatively, model-free methods estimate both shape and pose jointly, as in SAM 3D [3, 6, 27, 38, 84, 121]. However, prior approaches are typically restricted to tabletop robotics, streets, or indoor scenes where objects rest on a supporting surface. In contrast, our approach estimates both shape and pose for a broad range of object types across diverse scenes.

3D datasets. Sourcing 3D annotations is challenging: the modality itself is complex, and the specialized tools required are hard to master. Anecdotally, modeling a 3D mesh from a reference image can take an experienced artist hours (Sec. E). Instead, existing 3D datasets (*e.g.*, ShapeNet [11], Objaverse-XL [17]) primarily consist of single synthetic objects; without paired real-world images, models can only learn from rendered views. In the real-world domain, existing datasets are small and mostly indoors [26, 42, 72, 78, 90], or procedurally generated [12, 16, 76]. Models trained on such constrained data struggle to generalize.

Post-training. While post-training began with a single supervised finetuning stage [29, 102], strong pretraining [7] made alignment much more data efficient [37], enabling iterative preference-based alignment like RLHF [71] and online DPO [75, 91]. When post-training must provide a strong steer, self-training methods offer denser supervision–leveraging the model itself to generate increasingly high-quality demonstrations, rather than relying solely on preference signals [2, 20, 34, 124]. SAM 3D employs self-training to bridge the synthetic→real domain gap and break the data barrier for 3D perception; most closely resembling RAFT [20], but also incorporating preference tuning.

Multi-stage pretraining. Modern pretraining increasingly employs multiple training stages. Early work on curricu-

lum learning [4] provided a basis for staged data mixing in pretraining, with higher-quality data coming later [31, 68]. Abdin et al. [1], Li et al. [50] show that mixing synthetic/web curricula can achieve strong performance at smaller scales. Increasingly, additional mid-training stages are used for capability injection, such as context extension [31] or coding [81], and recent work finds that mid-training significantly improves post-training effectiveness [48, 101]. SAM 3D introduces pretraining and mid-training that can generalize for 3D.

B. Data Annotation Engine Details

B.1. Stage 1: Image and Object Candidate Sourcing

Image sources. To promote generalization across diverse real-world scenes, we expanded our domain coverage by sourcing images from multiple datasets. These include large-scale web-sourced imagery (SA-1B [44], MetaCLIP [115]), video data capturing everyday environments (SA-VI [51]), egocentric video datasets (Ego4D [32], Ego-Exo4D [33], AEA [58], AEO [87], Nymeria [59]), and domain-specific collections such as food (Food Recognition [5]) and driving scenes (BDD100k [123]).

We first filter out images with low resolution, severe blurriness, low contrast, or noticeable artifacts to ensure high-quality visual inputs that are representative of real-world scenarios. Next, we employ visual-language models for object recognition to generate object-level annotations for each image. Images containing only uninformative backgrounds (*e.g.*, ground, sky, ocean) without salient 3D objects are subsequently removed from the dataset.

For each object description, we employ a referral segmentation model to visually ground the object, followed by human annotator verification or refinement of object masks. We discard low-quality masks, masks covering multiple objects, or partial masks that do not capture a distinct object part. This ensures that each retained mask corresponds to a clearly indexable single object instance with sufficient granularity.

2D object selection In addition to the objects manually selected and masked by annotators, we also supplement our object mask inputs with segmentation masks sampled from pre-existing datasets. Besides saving annotation time, this strategy gives us more fine-grained control over the object distribution of the input masks, as object distributions are difficult to enforce on a per-image or per-annotator basis. To ensure a broad coverage of object categories, we adopt two complementary sampling strategies. First, we construct a 3D-oriented taxonomy by carefully merging and modifying the LVIS [35] $1,200$ object categories, emphasizing representations of 3D geometry. For example, different dog breeds are grouped together due to their similar underlying 3D structures, regardless of color, texture, or size. Second, we incorporate human annotator input to identify additional salient objects that may fall outside the taxonomy or are difficult to describe using text alone.

We retain object category labels and continuously monitor the distribution of objects passing through our data engine. To balance throughput and efficiency, we employ a curriculum-inspired sampling strategy, progressing from simple to increasingly complex geometries. Specifically, we begin with rigid objects of simple shapes (*e.g.*, balls, cylinders), transition to more structurally complex objects (*e.g.*, tools, buildings) and ultimately include non-rigid and highly deformable objects (*e.g.*, animals, humans, clothing). The sampling distribution is adaptively adjusted to reflect the evolving dataset composition, with particular emphasis on gradually expanding coverage of long-tail object categories. Through this strategy, we're able to source 850,000 unique object instances from 360,000 images, with annotations covering a wide range of object categories.

Texture MITL-3DO. The MITL-3DO dataset for texture is separate from the dataset for shape and layout, but is collected in a similar fashion. The images are sourced from SA-1B [44], and we additionally sample a dataset of examples with higher aesthetics – objects with minimal occlusion and high brightness, contrast, colorfulness, sharpness, and aesthetic score – to seed the model with higher-quality texture annotations. We found the high-aesthetics dataset to further improve human preference rate (see "AES" preference win rate in Figure 16).

B.2. Stage 2: 3D Model-in-the-Loop Suite

3D shape model suite. 3D shape generation is beyond the capabilities of the average human annotator, and it is a time-consuming process even for trained specialists (see Section E.1). Thus, in order to scale shape generation in our annotation pipeline, we instead convert the task to one of verification. We achieve this by employing a diverse set of 3D models to generate shape predictions for each object, asking annotators to pick and grade the best of N options. The sources of 3D shapes in our annotations include the following:

- **Retrieval**: The nearest 3D object is retrieved from a shape object library (pretraining data) using both image- and text-based similarity. For text similarity, we compare visual object descriptions; for image similarity, we compute the distance between CLIP embeddings. While this retrieval approach is nearly guaranteed not to provide an exact 3D reconstruction, it can provide a high quality mesh with matching semantics, particularly when model-generated 3D shapes fail entirely.
- **Text-to-3D generation**: A text-to-3D generative method produces 3D object meshes based on a textual descriptions.

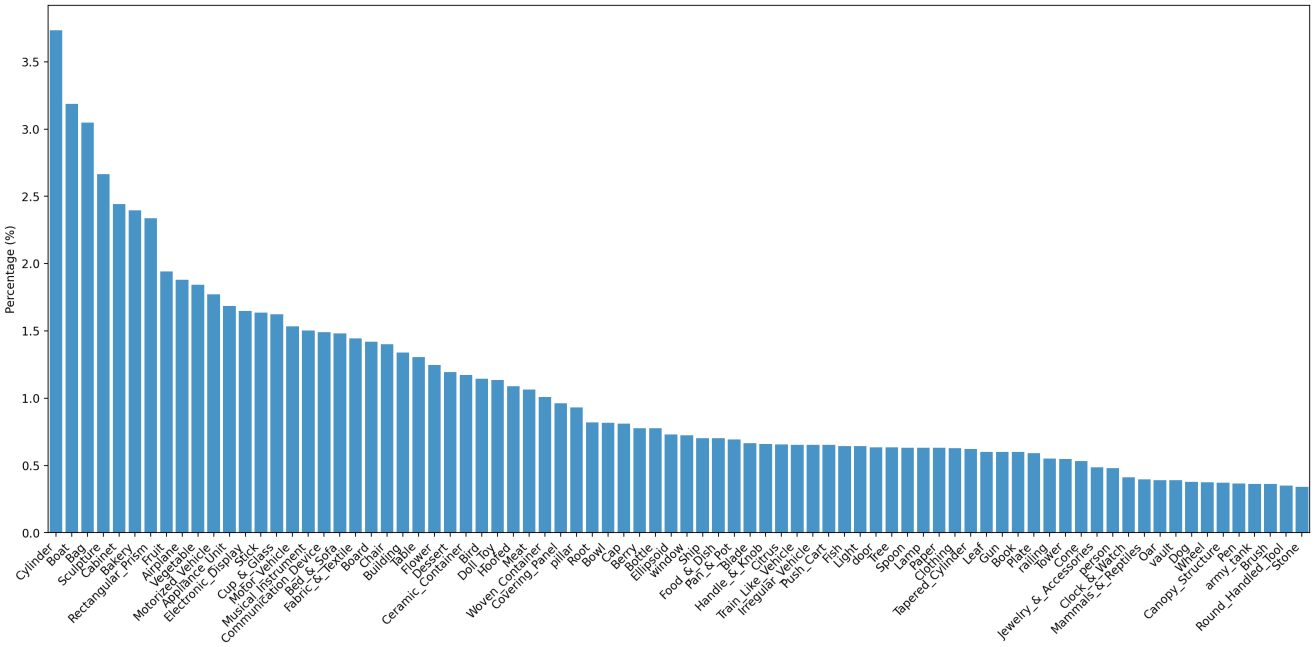

Figure 10. **Category distribution of SAM 3D training data**. The plot above shows the distribution of the top 80 object categories, which includes a long tail.

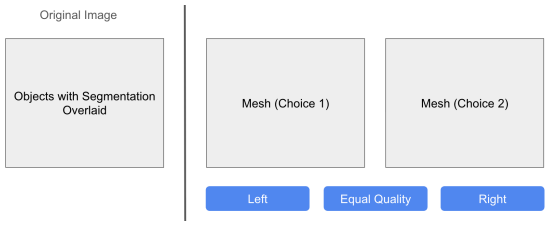

Figure 11. **Stage 2 UI Sketch.** Annotators can only choose between options; they cannot directly edit the meshes or textures.

This approach can be helpful when image-conditioning is challenging due to clutter or occlusion, but human recognition can still identify the object.

- **Image-to-3D generation**: Image-to-3D methods, including our own SAM 3D checkpoint, generate 3D objects in the form of point clouds or meshes, conditioned on the image input. When successful, this tends to produce examples that go beyond semantic matches and better respect the object's physical appearance in the image. However, lack of robustness to occlusion or clutter can negatively impact the results.

3D texture model suite. For texture generation, we utilize image-to-3D models, multi-view texture generation models, and our own SAM 3D checkpoints. All texture candidates are generated using shapes produced by the SAM 3D Geometry model, ensuring that texture models have the best chance of success, even in cases of heavy occlusion.

Stage 2 Selection procedure. The annotators select the best-of-N candidates by making a series of pairwise comparisons (see Figure 11). For each object, the annotator is initially presented with two candidates to compare and is asked to pick from the following three choices: "Left" (is better), "Right" (is better), or "Equal Quality". Because the options are in 3D, we by default automatically rotate the objects on a turntable, but annotators are free to rotate the objects as they wish, or zoom the camera. After making a selection, the non-selected option is replaced by a new candidate; if "Equal Quality", we randomly choose which candidate to keep. The selection procedure continues until all candidates have been shown. We randomize the order in which candidates are presented to the annotator, to prevent biases due to order from affecting the selection process.

After the best candidate is identified in the selection process, annotators are asked to rate the mesh against a predefined quality bar α. Examples meeting the bar will become candidates to enter Stage 3 for alignment, while examples under the bar will become negative examples for preference alignment or considered as candidates for manual mesh generation in Stage 2.5.

B.3. Stage 2.5: 3D Artist Mesh Details

When the 3D model-in-the-loop suite fails to generate an acceptable mesh for a particular sample, the aforementioned preference-based annotation approach is unable to provide the data needed to improve the model for such objects. To overcome this data distribution blind spot, we work with a

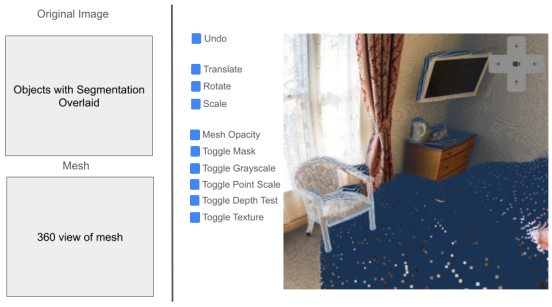

Figure 12. **Stage 3 UI Sketch.** The UI supports annotators in directly placing the object in the 2.5D pointcloud.

team of 3D artists to build meshes for such hard meshes. Given the high cost of specialized 3D artists, we seek to maximize their value by ensuring each object sent to the 3D artists represents a genuine failure case that cannot be resolved by the data engine alone. To maximize the value of this investment, we develop a refined labeling framework that categorizes failures into common types: *e.g.*, complex geometry, occlusion, transparency, and small object size. We balance sampling across these categories. In addition, we employ clustering techniques over images, 3D latents, and object semantics to deduplicate candidates, ensuring that one or a few representative samples per group suffices for effective coverage in data sampling.

Additional details on the data collection process for meshes created by 3D artists can be found in Section E.1, which employed a similar mesh creation process by the artists, but with more intentional curation of inputs.

B.4. Stage 3: 3D Mesh Alignment

We collect object pose annotations by aligning meshes from prior stages to a scene point cloud derived from the input image. To make this accessible to generalist annotators, we designed and implemented an annotation tool which allows the annotator to manipulate 3D meshes to align to a 2.5D point cloud pre-computed by an off-the-shelf depth estimator. Annotators can use either keyboard or mouse to rotate, translate, and scale the meshes so that the mesh is accurately anchored to the 2.5D point cloud. We also provide additional functions including (a) mesh visibility toggle, (b) target indicator toggle, (c) point cloud size adjustment, (d) control visibility toggle, (e) undo, (f) camera view reset and pre-defined view, and (g) mesh IOU indicator as shown in Figure 12.

B.5. Annotation Statistics

- Stage 1: Annotators on average spend 10 seconds to segment a single interesting object. We utilize SAM [44] as a tool to assist in segmentation.
- Stage 2: Annotators on average spend 80 seconds to select the best candidate shape/texture from 6-10 candidate meshes from variable sources.
- Stage 3: Annotators on average spend 150 seconds to anchor and orient the matched 3D shape to the 2.5D point cloud.
- Over the lifetime of the project (including development), our MITL data engine yields 3.14 million trainable shapes, 1.23 million samples of layout data, 100K trainable textures, and over 7 million pairwise preferences.

B.6. Core Alignment Algorithm

B.6.1. Basic Algorithm

Algorithm 1 shows the core alignment algorithm, used for all texture annotations and most shape annotations (MITL-3DO). During each collection step, we generate a set of predictions from the current model, and ask annotators to rank and verify these predictions. Generalist annotators can only choose between model outputs and accept/reject; they cannot edit. We maximize the probability of a successful annotation at each iteration by ensembling multiple models and combining multiple models with human preferences into an *expert* annotator.

The learning efficiency of the alignment, or the "speed of the data flywheel", is controlled by two factors:

- **Amplification factor**: The size of the performance gap between current model and the expert annotations at each iteration
- **Stepwise efficiency**: How closely the new model approximates the previous expert from the previous iteration

The former induces an upper bound on the new policy's performance at each iteration, while the latter describes how closely we approach that upper bound – similar to Expert Iteration [2].

B.6.2. Training Intuition

Our goal in post-training (see Algorithm 1) is to align the model to match human preference on the distribution of *all* possible real-world objects.

The core algorithm in our data engine generates samples by asking humans to select viable samples from a set of candidate generations. Challenging inputs often result in no viable candidate generations and thus never get selected by humans. However, at any current time our model is usually good on some parts of the data distribution, but not on other parts, as shown in the cartoon below:

The intuition behind the data engine in SAM 3D is that these green islands of reliably good performance correspond to high-density parts of the training data [67, 69], and the approach in SAM 3D is that we want to push out from these islands of reliably good generations into the "tail" of the distribution, demarcated by yellow and white background in the cartoon above. The yellow parts of the distribution are challenging for the model, but near enough to the blue islands,

Algorithm 1 SAM 3D Basic Alignment (Texture, Shape)

Require: Base model π_0, quality threshold curriculum α_k, ensemble size N
Ensure: Aligned model π_K

// Let $d = (I, M, S, T, R, t, s)$ denote a demonstration (i.e., a training sample)
for $k = 1$ to K **do**
  // Collection Step: Generate demonstrations via expert policy
  Initialize $\mathcal{C}_k \leftarrow \emptyset$ ▷ The dataset collected during iteration k
  for $(I, M) \sim p(\mathbf{I}, \mathbf{M})$ **do**
    $\tilde{\pi}_k \leftarrow \text{Amplify}(\pi_{k-1})$ ▷ Amplify current policy via model ensemble and best-of-N search
    Sample $\{d_i\}_{i=1}^N \sim \tilde{\pi}_k(I, M)$ ▷ Generate N candidate demonstrations from expert policy
    $d^*, r \leftarrow \text{HumanRank}(\{d_i\}_{i=1}^N)$ ▷ Humans select best candidate via pairwise comparisons
    $\mathcal{R} \leftarrow \{d_i : i \neq \arg\max\}$ ▷ Store rejected candidates for preference learning
    $\mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{(d^*, r, \mathcal{R})\}$ ▷ Collect chosen demonstration with rating and rejections
  end for
  // Update Step: Train on aggregated high-quality demonstrations and preferences
  $\mathcal{C} \leftarrow \{(d^+, \mathcal{R}) : (d^+, r, \mathcal{R}) \in \bigcup_{i=1}^k \mathcal{C}_i, r \geq \alpha_k\}$ ▷ Aggregate and filter by quality
  $\mathcal{D} \leftarrow \{(d^+, d^-) : (d^+, \mathcal{R}) \in \mathcal{C}, d^- \in \mathcal{R}\}$ ▷ Create preference pairs for DPO training
  $\pi_k^{\text{SFT}} \leftarrow \arg\min_\pi \mathbb{E}_{(d^+,d^-)\sim\mathcal{D}}[\mathcal{L}_{\text{CFM}}(\pi; d^+)]$ ▷ Supervised finetuning
  $\pi_k \leftarrow \arg\min_\pi \mathbb{E}_{(d^+,d^-)\sim\mathcal{D}}[\mathcal{L}_{\text{DPO}}(\pi, \pi_k^{\text{SFT}}; d^+, d^-)]$ ▷ Align with preferences
end for
return π_K

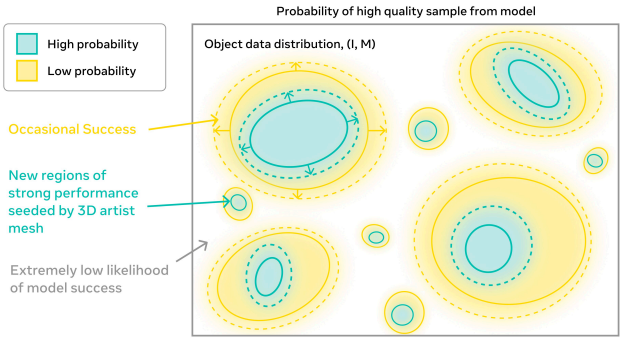

Figure 13. **Simplified cartoon depiction of data engine improvement**: The diagram depicts model sample quality (color) across the real-world distribution of images and masks. During training, the model begins by doing well (teal) on common categories and simple objects (chairs, bottles, signs, cars). Our goal is to both improve accuracy and robustness on these easy examples (**teal**), and then push the model to improve performance on less common objects (**yellow**) in the tail of the probability distribution.

that we can *occasionally* generate satisfactory annotations, but it requires humans to go through many samples.

This can create a chicken-and-egg problem where, for the model to become good, it must already be capable of produce a good generation; at least some of the time. For examples that are so challenging that the probability of success is extremely low (white), the model has no hope and we ask human 3D artists (Section B.3) to provide supervision in this part of the data distribution, in order to seed new islands.

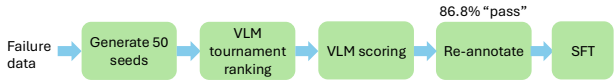

Figure 14. **Reward model data recovery pipeline**. The diagram shows how we use reward mdoels to increase N in best-of-N search to improve the chance of a successful annotation on challenging tail inputs. We use both a VLM and also DPO implicit reward as reward models.

B.7. Increasing Amplification Factor with Search

B.7.1. Best-of-N Search with Reward Models

Qualitatively, however, we observe that re-visiting some (yellow) inputs with a large number of seeds, our model can sometimes still yield a few good generations (*e.g.*, food can take around $\sim$ 50 seeds to reliably generate a successful mesh in Table 12). This suggests that increasing N in the best-of-N rejection sampling can potentially allow us to obtain annotations for challenging inputs, which would be difficult to source otherwise. Doing so would allow us to rapidly "push into the tail", increasing the convergence speed of the alignment algorithm in Algorithm 1. However, the primary impediment to increasing N is that, at some point, there are too many choices for a human to compare. This linearly scales the annotation time of preference data collection, and the selections themselves become noisier and more random due to choice overload [19].

To address this challenge, we explored using learned re-

ward models to perform a first pass in order to surface a smaller number of candidates for humans to then choose between. Figure 14 shows a pipeline to perform reward-ranked best-of-N search that increases the yield of successful annotations on challenging inputs. We first run 50 generations with different initial noise, and use the reward model to perform tournament-style ranking, and then pass the winning candidate to human annotators for ranking and verification (as in Stage 2).

We find that this approach indeed helps to recover some of this otherwise difficult data. For example, by scaling the best-of-N from $N = 2$ to $N = 50$ to recover samples that were originally discarded, improving the yield from 0% (since these were originally failures) to 86.8%. In particular, we observe significant increase in the proportion of successful annotations coming from challenging categories. The *food* category improves $9\times$ from 4% in the original annotated distribution to 36%. We show the experiments with resulting model performance, as well as ablations using VLMs instead of DPO implicit reward models, in Section F.7.

B.8. Relationship to Self-Training Approaches

The data engine in SAM 3D can alternatively be viewed as an online alignment algorithm similar to RLHF [71] or related self-training methods. Under this interpretation, the generative model q is a policy and the data collection step is a policy evaluation; collecting demonstrations $\mathcal{D}+$ and preferences $\mathcal{D}^+/\mathcal{D}^-$ through the interaction with the environment (annotators). The model improvement step simply updates the current policy using both finetuning and DPO.

This reframing helps make the relationship to existing work more clear. The most similar learning algorithm to our data engine is Expert Iteration (ExIt) [2]. As in ExIt, each iteration starts with a current policy, that we amplify using additional information into an expert policy, and we use this expert policy to generate supervision for imitation learning. Unlike ExIt, which uses purely imitation learning, we use humans-as-verifiers to select which samples to train on, and we make use of additional preference signals as reward signal (Section 3.2.2). However, there are also notable differences in type of supervision that can be used and the amplification steps. Our expert policy amplification step uses a model ensemble instead of only tree search with a value function, and we use preferences in the update step to better align to the task.

Algorithm 1 uses reward ranking, similar to RAFT [20] and RFT [124], although the alignment algorithm in SAM 3D adds explicit expert policies/ensembles and leverages preference supervision.

C. Pretraining and Mid-Training Data Details

C.1. Iso-3DO Data Filtering

For the Iso-3DO data used for pretraining, the quality of the 3D meshes can vary substantially, and not all samples exhibit high-fidelity geometry. Such examples can ultimately prove harmful to model pretraining, even at scale. One way to filter data is by an aesthetic score filter, as employed by Xiang et al. [112], which primarily emphasizes visual and textural appeal. We employ a similar filter process for the Texture & Refinement model.

However, this filter does not necessarily capture the geometric quality of a training data. Therefore, we develop a rule-based filtering strategy on shape to curate the pretraining data for the Geometry model, removing data with following characteristics:

- **Overly simplistic geometry**, characterized by extremely small volumes (*e.g.*, near-degenerate point-like structures) or minimal normal direction variation (*e.g.*, flat, sheet-like surfaces).
- **Structural outliers**, which includes meshes containing spatial outliers: isolated points or fragments that deviate significantly from the primary 3D structure.

C.2. Render-and-Paste Data Pipeline

We define the Render-Paste approach as follows: Given a natural image and an object instance defined by its mask segment, we replace the object in the image with a synthetic 3D object drawn from the same synthetic sources used in Iso-3DO. The size and position of the 3D object are determined using the 2D object mask together with a pointmap produced by a single-image depth estimator, which also guides the object's visibility and occlusion to obtain a natural appearance in the final rendering. By nature of starting from a synthetic 3D object first, the resulting data (which we refer to as *RP-3DO*) has excellent 3D ground truth precision and pixel-alignment compared to subsequent data sources in our training pipeline, which much try to reconstruct 3D from partial 2D information as part of the data annotation process.

In the following sections, we introduce three variants of Render-Paste that differ along two axes: pose information and semantic relevance.

- Section C.2.1: *Flying Occlusions* **(FO)** inserts randomly oriented synthetic objects without pose information, resulting in pose-unaware but semantically loose composites.
- Section C.2.2: *Object Swap – Random* **(OS-R)** determines object scale and translation from masks and pointmaps, while using a random rotation and object. Beyond simple replacement, the incorporation of depth ordering provides meaningful visual cues for object size and spatial placement, yielding pose-aware but not fully aligned insertions with moderate semantic relevance, higher than in the Flying Occlusions setting.

- Section C.2.3: *Object Swap – Annotated* **(OS-A)** replaces the original object using the annotator-provided ground-truth scale, translation, and rotation, producing fully pose-aligned and semantically matched renderings.

C.2.1. Flying Occlusions (FO)

The aim of this dataset is to build invariance to occlusion and size variations that commonly occur in real-world scenarios—and to enable the model to leverage full image context instead of only object-centered crops—we construct a dataset of natural images with blended synthetic 3D objects. Inspired by Flying Chairs [21] and FlyingThings3D [62], we name our first variant Flying Occlusions, reflecting its use of freely inserted synthetic objects.

Each training example consists of a natural image onto which we composite two rendered 3D objects: an *occluder* and an *occludee*. For each pair, we also compute the final visible mask of the occludee after occlusion. To generate each training sample, we randomly pair a selected object with an occluder object. Given the mask of the selected object M_{obj} and the mask of the random occluder M_{occluder}, each corresponding to the full mask of the respective object, the visible mask is defined as $M_{\text{vis}} = M_{\text{obj}} \odot (1 - M_{\text{occluder}})$, where $\odot$ denotes the element-wise (Hadamard) product. To ensure a reasonable degree of occlusion, we enforce $0.1 \leq |M_{\text{vis}}|/|M_{\text{obj}}| \leq 0.9$. In addition, samples with insufficient visibility are filtered out by requiring $|M_{\text{vis}}|/|I| \geq 0.2\%$, where $|I|$ is the total number of pixels in the image. Here, $|M|$ denotes the sum of all elements in M (*i.e.*, the total number of pixels with value 1 if M is a binary mask).

Finally, to prevent the model from always predicting the occluded object, in one third of the samples, we treat the selected mesh as the occluder. In these cases, the mask of the selected mesh is complete. In total, we have 55.1M sample with 2.87M unique meshes and 11.17M unique images.

C.2.2. Object Swap – Random (OS-R)

To enhance robustness to variations in object location and scale, we propose Object Swap – Random (OS-R), a depth-aware render-paste strategy that replaces an object in a natural image with a randomly selected synthetic mesh.

Given a natural image I, mask M, and random object mesh S, we synthesize a new training tuple $(I', M_{\text{vis}}, S, R, t, s)$. We first predict the 2.5D scene pointmap and identify the 3D centroid and bounding box of the target object. The original object is removed via inpainting, and we then insert a random synthetic mesh S at the computed centroid t with a random 3D rotation R. The mesh scale s is determined by fitting the mesh to the original object's 3D bounding box.

We complete the process by re-rendering the new image I' with a z-buffer check. We render the new mesh into the inpainted image such that only pixels not occluded by existing scene geometry are visible, forming the visible mask M_{vis}. We filter samples with insufficient visibility ($< 20\%$ visibility) and update the pointmap P by projecting the unoccluded surface points of the new mesh, using M_{vis}.

To ensure the dataset provides sufficient visual cues for estimating translation and scale, we use heuristics to replace only objects that are partially occluded or supported along the bottom, which provides depth ordering and T-junction cues, respectively. We verify these cues by trying to find occlusion boundaries: we sample points on opposite sides of the mask border, and if the outer pixel is significantly closer to the camera than the inner pixel, we consider this part of the boundary occluded. A sample is retained if it meets one of two conditions: (1) *physical support*, where the background is closer to the camera along the bottom 10% of the object (indicating it rests on a surface), or (2) *partial occlusion*, where foreground elements occlude at least 10% of the total object perimeter. This process yields 5.95M training samples composed of 2.38M unique meshes and 1.20M unique images.

C.2.3. Object Swap – Annotated (OS-A)

In addition to the *Object Swap – Random* variant, we construct a complementary render-and-paste setting, which we refer to as *Object Swap – Annotated* (OS-A), which performs an in-place replacement of a real image with a rendered human-annotated object. The motivation for this dataset is to enable Texture & Refinement training that faithfully preserves pixel-aligned correspondence between the rendered mesh and the visual appearance of the target object in the image.

This approach closely follows the *OS-R* pipeline, with key distinctions arising from the use of human-annotated data in MITL-3DO. Specifically, each training sample is generated using an image from a curated MITL-3DO subset, where the initial object mask, selected mesh S, object placement (translation t, rotation R, and scale s), and target pose are all sourced from human annotations provided in the MITL-3DO dataset. The selected mesh for each object is chosen by annotators as the best available match to the object's appearance in the image. During rendering, lighting conditions used for rendering are carefully matched to those in the training data preparation, ensuring consistent brightness and appearance across the dataset. We used a subset of the MITL-3DO shape-preference annotations, yielding 0.4 million training samples from this render-paste process.

C.3. Lighting for Texture Data

For Iso-3DO and RP-3DO (FO), we randomize the lighting (*i.e.*, direction and intensity) applied on the input images, and use ambient lighting when rendering views used to computing the target latents. Qualitatively, such data processing encourages the model to predict "de-lighted" textures without baking in strong directional shading or specular highlights from the input render. We verify that this is preferred by

humans through preference tests (see "Lighting Aug" preference rate in Figure 16).

D. Details on Model Training

The following sections outline the details of the Geometry and Texture & Refinement models, including architecture, training objective, and training hyperparameters.

D.1. Architecture Details on Geometry Model

We employ a latent flow matching model. For shape, it denoises the 64^3 voxels in the latent space of a coarser $16^3 \times 8$ representation, following Xiang et al. [112]. For layout, we perform denoising directly in the parameter space (R, t, s), as their dimensionality is small. Additionally, we introduce modality-specific input and output projection layers to map both the shape and layout parameters into a shared feature space of dimension 1024, and subsequently project them back to their respective parameter spaces. This results in a total of 4096 tokens for the shape and 1 token for R, t, s, respectively, as input to the Mixture of Transformers (MoT). The MoT architecture comprises two transformers: one dedicated to the shape tokens, and a second whose parameters are shared for the layout parameters (R, t, s), as shown in Figure 2.

The MoT design allows independently training of some modalities while maintaining performance on others (*e.g.*, fine-tune shape or layout only), thanks to the structured attention mask illustrated in Figure 2. This proves helpful when training on datasets that contain labels for only one modality (*e.g.* shape-only), and when freezing shape capabilities and finetuning just for layout. At the same time, MoT still allows for information sharing during the forward pass, through the joint self-attention layers for cross-modal interaction. This shared context is critical for self-consistency: notably, rotation is only meaningful when anchored to the predicted shape.

D.2. Pretraining & SFT Objective: Conditional Rectified Flow Matching

SAM 3D is trained to jointly generate multiple 3D modalities using rectified conditional flow matching [57]. Given an input image I and mask M, the Geometry model optimizes the following multi-modal flow matching objective:

$$\mathcal{L}_{\text{CFM}} = \sum_{m \in \mathcal{M}} \lambda_m \mathop{\mathbb{E}}_{\tau, \mathbf{x}_0^m} \left[\| \mathbf{v}^m - \mathbf{v}_\theta^m(\mathbf{x}_\tau^m, c, \tau) \|^2 \right] \quad (1)$$

where $\mathcal{M} = \{S, R, t, s\}$ denotes the set of prediction modalities (shape, rotation, translation, scale), $c = (I, M)$ contains the conditioning modalities (image, mask), and $\mathbf{v}_\theta^m$ is the learned velocity field for modality m at the partially noised state, $\mathbf{x}_\tau^m$.

We want to learn to generate clean states $\{\mathbf{x}_1^m\}_{m \in \mathcal{M}} \sim p(\mathcal{M}|c)$, and during training these are the ground-truth 3D annotations for each modality. Then, the target probability path at time $\tau \in [0, 1]$ is a linear interpolation $\mathbf{x}_\tau^m = \tau \mathbf{x}_1^m + (1 - \tau)\mathbf{x}_0^m$ between the target state $\mathbf{x}_1^m$ and intial noise state $\mathbf{x}_0^m \sim \mathcal{N}(0, \mathbf{I})$. As a result, the target velocity field is the gradient of this linear interpolation $\mathbf{v}^m = \dot{\mathbf{x}}_\tau^m = (\mathbf{x}_1^m - \mathbf{x}_0^m)$. λ_m is simply a per-modality weighting coefficient.

The Texture & Refinement model optimizes analogous flow-matching objectives using SLAT features. We train both models using AdamW (without weight decay), and training hyperparameters such as sampling and learning rate schedules, EMA weights are in Section D.7.

D.3. Preference Alignment Objective: DPO

For preference alignment in Section 3.2.2, we follow Diffusion-DPO [97] and adapt to flow matching as follows: given the same input image and mask c, we sample a pair of 3D output (x_0^w, x_0^l) based on human preference, where x_0^w is the preferred option and x_0^l is the less preferred. Our training objective is:

$$\begin{aligned} \mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\substack{I \sim \mathcal{I}, \\ (x_0^w, x_0^l) \sim \mathcal{X}_I^2 \\ \tau \sim \mathcal{U}(0,T) \\ x_\tau^w \sim q(x_\tau^w \mid x_0^w) \\ x_\tau^l \sim q(x_\tau^l \mid x_0^l)}} \left[\log \sigma \left(-\beta T w(\tau) \cdot \Delta \right) \right] \end{aligned} \quad (2)$$

where (3)

$$\begin{aligned} \Delta = &\| \mathbf{v}^w - \mathbf{v}_\theta(x_\tau^w, c, \tau) \|_2^2 - \| \mathbf{v}^w - \mathbf{v}_{\text{ref}}(x_\tau^w, c, \tau) \|_2^2 \\ &- \left(\| \mathbf{v}^l - \mathbf{v}_\theta(x_\tau^l, c, \tau) \|_2^2 - \| \mathbf{v}^l - \mathbf{v}_{\text{ref}}(x_\tau^l, c, \tau) \|_2^2 \right) \end{aligned}$$

where $\mathbf{v}^w$ and $\mathbf{v}^l$ are the target flow-matching velocities for x_τ^w and x_τ^l, and $\mathbf{v}_\theta$, $\mathbf{v}_{\text{ref}}$ are the learned and frozen reference velocity fields, respectively.

Implementation details. We apply DPO on shape prediction in the Geometry model and the predictions of the Texture & Refinement model. We use the preference data collected in Stage 2, where we remove the negatives from non-SAM 3D generations (*e.g.* retrieval-based methods or multi-view diffusion texture generations), since they are out of the distribution for SAM 3D.

D.4. Model Distillation Objective: Shortcut Models

For applications needing online 3D perception capabilities (*e.g.* robotics), model inference time is an essential consideration. In diffusion and flow matching models, the most straightforward way to improve inference speed is by reducing the number of function evaluations (NFE). However, naïvely decreasing the number of steps can significantly degrade performance. Instead, we employ flow matching distillation techniques to reduce the number of inference steps while minimizing impact to quality. Specifically, we

adopt the diffusion shortcut formulation from Frans et al. [25], which offers several advantages over previous consistency distillation approaches: (1) it is simple, avoiding multi-stage training and instability; and (2) the model supports two modes, allowing seamless switching back to the original flow matching inference, so a single model can serve both purposes. Unlike the original formulation, we do not train shortcut models from scratch. Instead, we initialize from fully trained checkpoints and further finetune them with the shortcut objective.

$$\mathcal{L}_S(\theta) = \mathbb{E}_{\substack{x_0\sim\mathcal{N}(0,I)\\ x_1\sim q(x)\\ \tau,d\sim p(\tau,d)}}\Bigg[\underbrace{\|\mathbf{v} - \mathbf{v}_\theta(x_\tau,\, c,\, \tau,\, d{=}0)\|^2}_{\text{Flow-Matching}} \tag{4}$$

$$+\, \underbrace{\|\mathbf{v}_{\text{consistency}} - \mathbf{v}_\theta(x_\tau,\, c,\, \tau,\, 2d)\|^2}_{\text{Self-Consistency}}\Bigg]. \tag{5}$$

where:

- $x_0 \sim \mathcal{N}(0, I)$: a Gaussian noise sample drawn from the standard normal distribution.
- $x_1 \sim p(x)$: a real data sample from the data distribution.
- x_τ: an interpolated sample between x_0 and x_1 at time step τ. (Defined earlier in the paper through the diffusion / flow matching path.)
- τ: the diffusion time (or noise level) at which the model predicts a local velocity or update step.
- d: the step size specifying how large a step the shortcut model should predict. $d = 0$ corresponds to flow-matching, $d > 0$ corresponds to consistency training.
- c: conditioning tokens.
- $p(\tau, d)$: the joint sampling distribution over diffusion times and step sizes used during training.
- $\mathbf{v}_\theta(x_\tau,\, c,\, \tau,\, d)$: the shortcut model parameterized by θ, taking as input the current sample x_τ, conditioning c, time τ, and desired step size d.
- $\mathbf{v}$: the empirical instantaneous velocity of the data flow used as the target for the flow-matching objective (corresponds to $d = 0$).
- $\mathbf{v}_{\text{consistency}}$: the self-consistency target, constructed by composing two steps of size d to form a reference for a single jump of size $2d$. Algorithm 2 describes how to construct $\mathbf{v}_{\text{consistency}}$.

We also distill CFG into the shortcut mode by using a fixed CFG strength of $w_{\text{CFG}} = 2$ for Stage 1 and CFG strength of $w_{\text{CFG}} = 1$ for Stage 2. The final model is finetuned for approximately 4K iterations using the same objective as in [25]: 75% flow matching and 25% shortcut. When shortcut mode is disabled, the model behaves identically to the original flow matching model. We initialize the step size embedder by setting the weights and bias of its final linear layer to zero, since, unlike the other parameters that have already been trained, these are new parameters introduced at the distillation stage. Figure 17 shows quantitative results and Figure 18 shows examples.

D.5. Texture & Refinement Training Details

We train the Texture & Refinement model following a multi-stage training paradigm analogous to that of the Geometry model (described in Section 3). Below we provide implementation details for the texture training stages.

VAE Training. Learning the inverse problem of image to texture map requires a strong alignment in training data between them. However, in previous work [112], renderings of meshes create artifacts like reflections and shadows; objects consistently having extremely dark bottoms is one such common example. To curate a cleaner set of ground truth data, we create our latent SLAT with a "de-lighted rendering" by using ambient lighting to minimize such artifacts. We use this to process all stages of data. We select a uniform lighting strength, based on computing the similarities between SAM 3D predictions and the original image using RP-3DO (FO).

Pretraining. We start with pretraining the model on Iso-3DO-500K, a partition of Iso-3DO data with high aesthetics. Training on such data allows the model to predict plausible, high-quality texture that is characteristic of many 3D asset generation models. To ensure robustness of the texture model on real-world, complex images, we must further train on increasingly challenging data, via additional training stages described below.

For pretraining data in Iso-3DO, we render conditional images from the 3D mesh. We introduce a random lighting augmentation, where for each view, we apply a random lighting setting in the rendering engine. We hope the model can learn to remove the lighting effects when generating textures.

Mid-training. During mid-training, we train on RP-3DO (FO and OS-A). It is starting at this stage where we additionally provide full image conditioning to the model, as we believe contextual cues can help the model predict plausible textures, especially when the object is heavily occluded. We show the effect of training on RP-3DO (FO), as well as the effect of further adding RP-3DO (OS-A) training data to this stage, in Figure 16.

We also introduce image data augmentation for RP-3DO (FO): **Mask** augmentation and **Blur** augmentation. The Mask augmentation randomly erode or dilute the input mask. This is designed to handle noise in mask at inference time (*e.g.* segmentation predictions from a model). The Blur augmentation applies a downsample operation on the image followed by an upsample operation. This is especially important to handle cases for motion blur and small objects in

Algorithm 2 Consistency Target Construction with CFG Guidance for Shortcut Model Distillation

Require: Current state x_τ, conditioning c, step size d, CFG weight w_{CFG}, teacher model $\mathbf{v}_\theta$
Ensure: Consistency target $\mathbf{v}_{\text{consistency}}$
// First shortcut step with CFG guidance
$\mathbf{v}_\tau \leftarrow \mathbf{v}_\theta(x_\tau, \varnothing, \tau, d) + w_{\text{CFG}}\big(\mathbf{v}_\theta(x_\tau, c, \tau, d) - \mathbf{v}_\theta(x_\tau, \varnothing, \tau, d)\big)$ ▷ Apply CFG to get guided velocity
$\tilde{x}_{\tau+d} \leftarrow x_\tau + d \cdot \mathbf{v}_\tau$ ▷ Take first step of size d
// Second shortcut step with CFG guidance
$\mathbf{v}_{\tau+d} \leftarrow \mathbf{v}_\theta(\tilde{x}_{\tau+d}, \varnothing, \tau+d, d) + w_{\text{CFG}}\big(\mathbf{v}_\theta(\tilde{x}_{\tau+d}, c, \tau+d, d) - \mathbf{v}_\theta(\tilde{x}_{\tau+d}, \varnothing, \tau+d, d)\big)$ ▷ Apply CFG at new state
$\tilde{x}_{\tau+2d} \leftarrow \tilde{x}_{\tau+d} + d \cdot \mathbf{v}_{\tau+d}$ ▷ Take second step of size d
// Compute consistency target from two-step trajectory
$\mathbf{v}_{\text{consistency}} \leftarrow \text{stopgrad}\left(\frac{\tilde{x}_{\tau+2d} - x_\tau}{2d}\right)$ ▷ Average velocity over combined $2d$ step
return $\mathbf{v}_{\text{consistency}}$

an image. These augmentation is carefully studied in Section F.2.

Supervised fine-tuning (SFT). In the SFT stage, the model is trained on MITL-3DO texture annotations, which includes the "aesthetic" samples described in Section B.1. We show the effect of scaling the MITL-3DO texture annotations by 2x in Figure 16, which improves human preference rate by 14.2%.

Preference optimization. Like the Geometry model, we run a final DPO stage to align the model with human preferences collected from the texture data engine. The effect of DPO on texture performance is shown in both Table 4 and Figure 16. We follow the same training objective described in Equation (4).

D.6. Texture & Refinement VAE

We make improvements over the original SLAT VAE design in Xiang et al. [112], where features are back-projected to all voxels, including those that are not visible (*i.e.*, occluded) from the current image. This original design choice leads to reduced sharpness in the reconstructed images. To address this issue, we back-project features only to voxels that are visible from each image, utilizing the depth information from that specific view. We call this VAE variant Depth-VAE. During training, we normalize the Kullback–Leibler (KL) regularization term by the active voxel count to prevent large objects from dominating the training loss. We also fine-tune the decoder for downstream needs, such as reducing the number of decoded Gaussians for faster inference.

D.6.1. Depth-VAE: Depth-Aware Feature Aggregation

To integrate depth information into patch-level features, we propose a depth-guided projection algorithm. Given a feature map $\mathbf{F} \in \mathbb{R}^{B\times C\times H\times W}$ and normalized 2D coordinates $\mathbf{U} \in [-1,1]^{B\times N\times 2}$, we sample features and aggregate them based on visibility, handling occlusions via a depth buffer.

Feature Sampling. For each coordinate $\mathbf{u}_i \in \mathbf{U}$, we extract the corresponding feature vector $\mathbf{f}_i$ from the DINO-V2 map $\mathbf{F}$ using differentiable bilinear interpolation (denoted as `GridSample`).

Occlusion Handling (Depth Filtering). To identify visible points, we construct a temporary depth buffer. We map the coordinates $\mathbf{U}$ to a discrete grid (P_x, P_y) and retain the minimum predicted depth $\hat{d}$ at each location to form a surface depth map $\mathbf{D}_{\text{surf}}$:

$$\mathbf{D}_{\text{surf}}(x,y) = \min_{i:(x_i,y_i)=(x,y)} \hat{d}_i. \tag{6}$$

We then resample this map at the original coordinates $\mathbf{U}$ to obtain the reference surface depth $\mathbf{d}_{\text{ref}}$. Note that if ground-truth depth is available, it is used in place of $\mathbf{D}_{\text{surf}}$.

Visibility Masking. We compute a binary mask $\mathbf{M}$ to discard occluded points. A point is considered visible if its depth $\hat{d}_i$ is within a tolerance τ of the reference surface $\mathbf{d}_{\text{ref},i}$:

$$\mathbf{M}_i = \mathbb{I}\left[\mathbf{d}_{\text{ref},i} > \hat{d}_i - \tau\right]. \tag{7}$$

We normalize this mask across the batch dimension (or views) to obtain weights $\tilde{\mathbf{M}}$.

Weighted Aggregation. The final depth-aware representation is the weighted sum of visible patch features:

$$\mathbf{F}_{\text{depth}} = \sum_b \tilde{\mathbf{M}}_b \odot \mathbf{f}_b. \tag{8}$$

D.7. Training Hyperparameters

We summarize training parameters in details in Table 5 for Geometry model and Table 6 for Texture & Refinement model.

We use a batch size of 6 per GPU for all training stages of the Geometry model, and epochs iterate over meshes. Pretraining is conducted on 512 A100 GPUs for 200 epochs. Mid-training on FO utilizes 320 A100 GPUs for 50 epochs, followed by further FO mid-training on 128 A100 GPUs for an additional 50 epochs, followed by OS-R midtraining

Training stage	Datasets	Condition input	Learning rate	Modality weights	# Meshes	# Images	# Samples	# Tokens
Pre-training	Iso-3DO	object-centric crop	10^{-4}	S=1.0, R=0.1	2.7M	64.8M	64.8M	2.5T
Mid-training	RP-3DO (FO)	full image	10^{-4}	S=1.0, R=0.1	2.87M	11.17M	55.1M	2.4T
	RP-3DO (OS-R), ProcThor	full image, pointmap	10^{-4}	S=1.0, R=0.1, t=1.0, s=0.1	2.38M	1.20M	5.95M	0.3T
SFT	MITL-3DO, Art-3DO	full image, pointmap	10^{-5}	S=1.0, R=0.1, t=1.0, s=0.1	0.6M	0.5M	0.6M	0.9T
Alignment	MITL preference	full image, pointmap	2.5×10^{-6}	S=1.0	88K	31K	44K	-

Table 5. **Detailed training hyperparameters for SAM 3D training stages (Geometry Model)**. This table extends Table 1 from the main paper with additional hyperparameter details.

Training stage	Datasets	Condition input	Learning rate	EMA	# Meshes	# Images	# Samples	# Tokens
Pre-training	Trellis500K	object-centric crop	10^{-4}	0.9999	350K	9M	10M	1.1T
Mid-training	RP-3DO (FO,OS-A)	full image	10^{-4}	0.9999	800K	2.4M	2.4M	1T
SFT	MITL	full image	10^{-5}	0.999	$\sim$100K	100K	100K	115B
Alignment	MITL preference	full image	10^{-6}	0.99	146K	73K	73K	-

Table 6. **Detailed training hyperparameters for SAM 3D training stages (Texture & Refinement Model).** This table extends Table 1 from the main paper with additional hyperparameter details.

on 256 A100s for 12 epochs. SFT is performed on 128 H200 GPUs for 100 epochs, training on data from our data engine as it becomes available. As this data leads to model improvements (and thus also improving the quality of data produced by the data engine), we raise our quality threshold α_k for keeping samples in our SFT training set; the final run uses an quality cutoff α_K that keeps 500K samples. DPO is performed on 128 A100 GPUs for 1 epoch.

For the Texture & Refinement model, we perform pre-training on 256 A100s for 245 epochs with a batch size of 4, followed by mid-training on 256 A100s for 80 epochs with a batch size of 4. For SFT, we use 192 A100s for 89 epochs and batch size of 4. Finally, DPO is conducted on 128 A100s for 2 epochs with a batch size of 3.

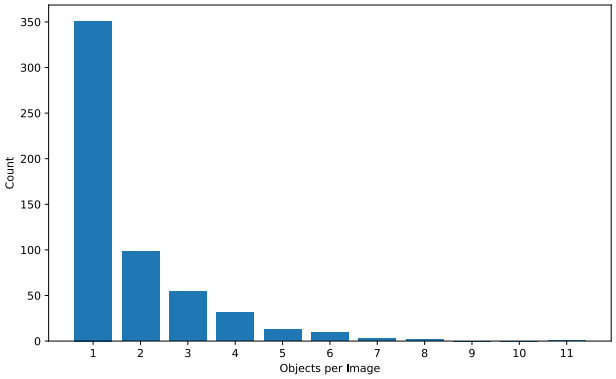

Figure 15. **Distribution of number of objects per image in SA-3DAO.** The number of objects follows a roughly power-law distribution.

E. Evaluation

Current evaluation benchmarks for visually grounded 3D object reconstruction fall short of capturing the complexity of the real world. Many rely on synthetic datasets [11, 17] where single objects are rendered in isolation, centered against a white background. This introduces a large visual gap with real-world evaluation conditions and the rich variation of real-world imagery. Efforts to move to real data mostly focus on indoor environments [42, 72, 88], but these benchmarks heavily skew toward furniture categories such as chairs and tables, limiting the diversity of objects that models must handle in practice. As a result, evaluations on these datasets do not reflect the challenges of natural environments, where objects are occluded, scenes are cluttered, scales vary, lighting conditions complicate appearance, and domains span far beyond indoor or synthetic scenes.

E.1. SA-3DAO: A New Benchmark for Real-World 3D Object Reconstruction

We introduce SAM 3D Artist Objects (SA-3DAO), a new benchmark designed to capture the diversity and complexity of real-world 3D perception. SA-3DAO consists of 1,000 untextured 3D objects, created from and carefully aligned to selected natural images capturing scenes spanning both indoor and outdoor environments, including parks, ski resorts, flea markets, parades and more. The benchmark covers a wide spectrum of object types: objects range from large, structured entities such as ski lifts and escalators, to everyday items like clothing, to rare and culturally specific objects such as tribal face masks. Crucially, the 3D ground truth are of high-fidelity created by professional 3D artists, who are tasked with producing accurate 3D shapes for the objects depicted in the input images. This combination of visual diversity, real-world context, and professionally crafted 3D ground truth makes SA-3DAO a comprehensive testbed for evaluating 3D object reconstruction models.

Collection details. We task professional 3D artists with recovering the shape of a target object from a single image, mirroring our model's goal of reversing the photographic transform (as described in Section 2.1). In other words, the

3D artists must create a whole 3D mesh that precisely aligns with the object's visible pixels in the image. Even under ideal settings, this requires contending with only partial information as the back side of the object is typically unseen, but many of the objects in SA-3DAO additionally have natural occlusions, or are small in size within the image; disentangling depth versus scale is often also challenging for a single image. To fill these information gaps, artists rely on recognition and context, using common-sense priors, physical plausibility, and assumptions of symmetry (when appropriate) to complete the meshes. The requirements imposed by this task are atypical to the normal 3D asset creation process that artists are more accustomed to, and efficient annotation requires learning a different mode of operation. After acclimation to the task, completion time per object mesh can vary considerably, ranging from up to 5 minutes for obvious objects with simple geometries to over 5 hours for more challenging cases; the median mesh in our dataset has 4751 vertices. Many of the images provided multiple objects with meshes from the 3D artists; we show the frequency distribution in Figure 15.

E.2. Human Preference Set

We further expand our evaluation suite to support more rigorous and domain-targeted assessments. While SA-3DAO provides a general and standardized way to measure progress, we want to also capture the challenges of settings where 3D perception is most critical, such as robotic manipulation and egocentric vision. To address this, we design a human preference set composed of images drawn from these domains of interest. This set enables evaluation through direct human judgment, providing insights that go beyond numerical metrics and capturing aspects of 3D perception that are important in embodied and real-world applications.

Domains. We design four human preference test datasets to comprehensively evaluate model capabilities across different scenarios.

- **SA-1B** [44]: We uniformly sample 1,000 image and object mask pairs, covering a diverse range of object categories. This set is intended to assess the model's generalization ability across varied object distributions, with particular emphasis on long-tail categories.
- **MetaCLIP** [115]: We select 1,000 samples where the object masks are of median or heavily occluded. This set evaluates the model's performance in reconstructing occluded objects, a common challenge in cluttered scenes.
- **LVIS** [35]: We densely sample 1,000 images containing between 10 and 30 objects per scene, and is designed to evaluate the model's transferability to out-of-domain data and to demonstrate its ability to capture physical properties within dense scene layouts.
- **Aria Digital Twin** [72]: We sample a smaller set of 40 video frames, with around 30 objects per scene. This dataset is intended to compare against baselines on scenes with highly accurate pointmaps, similar to those on which the baselines were trained.

Setup. Human preference evaluations are conducted through a structured sequence of pairwise comparisons. For each image and a masked object, annotators are first presented with two reconstructions (model "A" vs. "B") and asked to select the one that most accurately matches the object in the image. The chosen reconstruction is then compared against the output of a third model (model "C"), and this process continues iteratively until all candidate models have been compared. Through this series of binary decisions, the most accurate reconstruction is identified as the preference for that particular image. To ensure fairness and avoid bias, the order of comparisons is randomized and the identify of models are anonymized.

E.3. Evaluation Metrics

E.3.1. Shape Metrics Definitions

For shape evaluation on SA-3DAO, we first normalize the artist-created ground-truth mesh and the generated mesh independently into the range $[-1, 1]$. We then apply ICP alignment for each mesh pair before computing metrics. We uniformly sample 1M points from both meshes and report four complementary metrics that capture different aspects of geometric fidelity:

- **F-score @ 0.01:** Measures correspondence accuracy between the reconstructed and ground-truth points under a 0.01 threshold. We compute precision and recall between the two point clouds and report their harmonic mean. F1 evaluates how many points lie close to the ground truth and how completely the reconstruction covers the target shape.
- **Voxel-IoU:** Provides a coarse volumetric agreement score and is sensitive to gross errors in volume, silhouette, and topology. We voxelize both point clouds to 64^3 resolution and compute intersection-over-union over occupied voxels.
- **Chamfer Distance (CD):** Measures bidirectional nearest-neighbor distance between reconstructed and ground-truth point sets, highlighting fine-grained geometric deviation and penalizing missing or distorted regions.
- **Earth Mover's Distance (EMD):** Quantifies the minimal cost required to transport one point distribution to match the other. EMD is more stringent than CD, capturing global structural differences and enforcing bijective correspondence between distributions.

Together, these metrics provide a comprehensive view of reconstruction fidelity, from local accuracy to global shape consistency.

Moreover, to evaluate shape quality on the ISO3D dataset [22]—which consists of 101 in-the-wild synthetic

Model	Training Setup	SA-3DAO			
		F1 @ 0.01 (↑)	vIoU (↑)	Chamfer (↓)	EMD (↓)
SAM 3D	Full	**0.2344**	**0.2311**	**0.0400**	**0.1211**
	w/o training on MITL-3DO	0.2211	0.2220	0.0486	0.1338
	w/o training on Art-3DO	0.2027	0.2025	0.0578	0.1510
	w/o DPO on MITL-3DO	0.2156	0.2156	0.0498	0.1367

Table 7. **Train Stage Knockout.** The impact of training on MITL and 3D artist-generated data.

images without 3D ground truth—we measure perceptual similarity between the generated shape and the input image using ULIP [118] and Uni3D [126]. For each generated mesh, we uniformly sample $8,192$ surface points to form a point cloud representation, and compute cross-modal similarity between the point cloud features and image features.

E.3.2. Layout Metrics Definitions

To evaluate single-object pose and compare with existing methods, we employ standard 6D pose estimation metrics, and then define ICP rotation error below.

- **3D IoU:** Measures the overlap of 3D axis-aligned bounding boxes between predicted and ground-truth bounding boxes, using the intersection-over-union. Values range from 0 (no overlap) to 1 (perfect overlap).
- **ICP-Rot:** *ICP Rotation Error* is the residual rotation error (in degrees) after ICP alignment. Given predicted rotation R_{pred} and ground-truth rotation R_{gt}, the meshes are first posed, then ICP finds optimal alignment R_{ICP}, and **ICP-Rot** is the angle of this rotation in degrees.
- **ADD-S (Average Distance with Symmetry):** ADD-S [113] is the symmetrized average of the minimum point-to-point distances between predicted and ground-truth posed objects, normalized by object diameter:

$$\text{ADD}(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}|} \sum_{\mathbf{x} \in \mathcal{A}} \min_{\mathbf{y} \in \mathcal{B}} \|\mathbf{x} - \mathbf{y}\|_2 \tag{9}$$

$$\text{ADD-S} = \frac{\text{ADD}(\mathcal{M}, \mathcal{M}_{\text{gt}}) + \text{ADD}(\mathcal{M}_{\text{gt}}, \mathcal{M})}{2d} \tag{10}$$

  where $\mathcal{M}$ and $\mathcal{M}_{\text{gt}}$ are the predicted and ground-truth point clouds for the posed shape, and $d = \max_{\mathbf{x},\mathbf{y} \in \mathcal{M}_{\text{gt}}} \|\mathbf{x} - \mathbf{y}\|_2$ is the diameter of the ground-truth point cloud. The symmetrized formulation averages distances in both directions: from predicted to ground-truth and from ground-truth to predicted. Lower values indicate better pose accuracy.

  The original ADD-S metric definition [113] was designed for 6DoF pose estimation using a ground truth CAD model. In this case, when the predicted and ground truth shape are the same, the asymmetric and symmetric versions of ADD-S coincide. In SAM 3D we jointly estimate shape and pose, so generalize the metric to the symmetric version.
- **ADD-S @ 0.1:** A binary value per-sample indicating whether the ADD-S distance is less than 10% of the object's diameter.

Model	SAM 3D WR over baselines, SAM 3D shape			
	ISO3D	Preference Set	SA-3DAO	LVIS
Trellis	81.1	87.0	86.2	89.1
Hunyuan3D-2.1	63.8	87.0	86.2	89.1
Hunyuan3D-2.0	70.1	77.5	77.4	85.7
Unitex	83.3	84.7	84.5	88.3

Table 8. **3D Texture:** Preference results comparing SAM 3D to competing image-to-3D methods on ISO3D [22], Preference Set, and SA-3DAO. We compare to the recent Trellis [112], Hunyuan3D-2.1 [39], and Unitex [54], with the same shape of SAM 3D. The human preference rates represent preference for SAM 3D over each baseline approach.

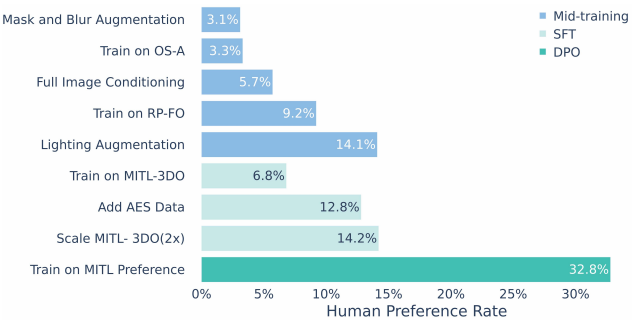

Figure 16. **Ablations for the Texture & Refinement model, grouped by training stage.** Percentages denote human preference rate for each ablation, over the model *without* the ablation.

F. Additional Ablations

F.1. Intermediate Training Stage Knockout

While Table 4 in the main paper shows the cumulative effect of adding different stages during training, Table 7 shows the impact of real-world data as intermediate stages. Knocking out any of these stages results in a notable drop in shape performance.

F.2. Texture Evaluations

Comparison to SOTA. We compare SAM 3D with existing methods on a holistic level (combined geometry and texture prediction) in Table 8. We compare against existing image-to-3D methods that predict Gaussians or textured meshes, including Trellis [112] and Hunyuan3D-2.1 [39]. We also conduct a texture-only comparison by providing SAM 3D geometry as input to the texture modules of the aforementioned baselines, with the addition of Unitex [54], a model that performs texture prediction given paired image and shape input.

We report human preference for SAM 3D over each baseline method on multiple datasets: ISO3D [22], Pref-

Model	3D IoU (↑)	ICP-Rot. (↓)	ADD-S (↓)	ADD-S @ 0.1 (↑)	2D IoU (↑)
SAM 3D	0.4837	14.4702	0.08265	0.7545	0.5143
SAM 3D (post-optim)	**0.5258**	**14.1460**	**0.07932**	**0.7617**	**0.6487**

Table 9. **Test-time optimization for layout:** Quantitative comparison of SAM 3D layout test-time optimization on Aria Digital Twin [72].

erence Set, SA-3DAO, and LVIS [35]. Results indicate that SAM 3D outperforms existing methods on the holistic image-to-3D task as well as texture estimation – this is due to the fact that the evaluated datasets often contain occlusion and clutter, which is a setting prior works struggle on.

Ablations for Texture Training. We conduct comprehensive studies on design choices for Texture & Refinement model (Figure 16) using annotator preferences on the Pref Set. We benchmark each component to the alternative model without the change. We remark a few themes here:

- Augmentation is very important, with lighting augmentation to be the most critical here. This is expected, given the Mask and Blur augmentations primarily focus on specific challenging cases (poor mask quality and low resolution inputs), so their effects get diluted in a holistic evaluation.
- RP-3DO data are critical and helps the model adapt to real world.
- Post-training data are critical, with significant gains coming from it. It demonstrates the effectiveness of our Data Engine, and DPO further amplifies the gains. In addition, sourcing specific type of data (AES) and scaling the data both show significant improvements.

F.3. Layout Test-Time Optimization

Render-and-compare is a longstanding popular approach for pose estimation [47, 103]: iteratively render the object's shape according to the most recently predicted pose, directly compare with the input pixels, and then adjust the pose prediction accordingly. This heuristic-based search can lead to fairly accurate results, even with weak initial pose samples ("proposals" from a base model). By contrast, SAM 3D operates in a feedforward manner, directly diffusing the object pose (rotation, translation, and scale) conditioned on the image features and, optionally, the scene pointmap. Notably, we do not include any sort of pixel-based loss objective in this process. However, SAM 3D's pose estimation can very naturally be used as a proposal for render-and-compare optimization. We include experiments showing the impact of this post-optimization process in Table 9, demonstrating that we can achieve further gains in pose metrics on ADT [72].

Specifically, we further optimize the layout proposals from SAM 3D by applying the layout to the generated objects, rendering and comparing for both masks and pixels, and backpropagating the gradients to refine the layout. We finally apply an automatic proposal-checking step, where the optimized layout is accepted only if its mask IoU exceeds that of the initial layout. As shown in Table 9, for the 554 accepted optimized samples out of the 1027 ADT instances, both the 3D layout metrics and the 2D mask IoU metric improve substantially, demonstrating the effectiveness of layout test-time optimization.

Representation	**Chamfer** (↓)	**ICP-Rot.** (↓)
Quaternion	0.0061	17.9585
6D Rotation	0.0074	15.5399
Normalized 6D Rotation	**0.0049**	**14.5946**

Table 10. **Rotation representation.** Ablation on the representation used during pretraining. We report Chamfer distances and ICP rotation error.

F.4. Rotation Representation

We compare different rotation representations while keeping all other architecture and training settings identical. Each model is trained on the pretraining dataset and evaluated on a held-out Objaverse test split of 216 samples. As shown in Table 10, switching from a quaternion representation to the 6D continuous rotation parameterization [127] yields a notable reduction in oriented rotation error, confirming that the 6D formulation provides a smoother optimization landscape more suitable for generative modeling. Further applying normalization to the 6D rotation vectors using the statistics over the training datasets leads to an additional improvement when training the flow matching models.

F.5. Pointmap Minimally Affects Shape

SAM 3D can condition on a 2.5D pointmap derived from sensor measurements (see Section 2.2) or from an off-the-shelf monocular depth estimation derived from the image itself, as is primarily used throughout this work. The latter is notable, as it also means that the current model can continue to benefit from future improvements of depth estimation methods. We observe that the pointmap minimally affects the shape performance: in a head-to-head preference test for shape on LVIS, the version of SAM 3D conditioned on pointmaps and the version without pointmaps are each selected 48% of the time.

F.6. Texture & Refinement *Depth-VAE* Comparison

Table 11 shows the result of the improvements made to the VAE used in the Texture & Refinement model; see Section D.6. We found that the depth feature significantly improves the perceptual quality of reconstruction, while scaling

Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Non-Depth VAE	30.65	0.9470	0.04776
Depth-VAE	30.87	0.9500	0.04579
Depth-VAE + scaling	**31.60**	**0.9547**	**0.04093**

Table 11. **Depth-VAE Ablations:** Effectiveness the depth-feature modification to the SLAT VAEs used in the Texture & refinement model. Results are evaluated on the entire GSO dataset.

	Tail Holdout		Epic Kitchens		SA-3DAO	
	Chamfer ↓	F1 ↑	Chamfer ↓	F1 ↑	Chamfer ↓	F1 ↑
SFT with $N = 2$	0.0059	0.39	0.0094	0.30	0.0083	0.26
SFT with $N = 50$ recovery	**0.0053**	**0.41**	**0.0090**	**0.32**	**0.0081**	**0.26**

Table 12. **Including reward-model-recovered data during SFT**, from the pipeline in Figure 14, improves model performance on challenging inputs, as seen in both Chamfer Distance and F1 score on the tail holdout set, Epic Kitchens [13], and SA-3DAO.

the training data further improves the reconstruction performance. We also notice that the enhancement primarily arises from the difficult scenarios for the non-depth VAE (when it performs poorly).

F.7. Data Engine: Increasing Best-of-N Search with Reward Models

Finetuning on data recovered using the reward-model-best-of-N pipeline improves model performance on various challenging inputs, such as the artist evaluation set, tail holdout set, as well as Epic Kitchens [13], as shown in Table 12. This demonstrates that further amplifying the expert policy in Algorithm 1 by increasing N in the best-of-N search can improve the robustness of the model in challenging categories, and suggests that improved test-time search can increase the alignment convergence speed of the data engine.

We found that both vision-language models (VLMs), and also the implicit reward models from our DPO stage [49] performed similarly in our case. In our testing, the VLM-as-reward model had 68.9% binary choice agreement with humans rater preferences, DPO agreed $\sim$ 65%, and two human annotators agreed $< 75\%$[1] of the time. Around 80% of the recovery data came from the DPO-as-reward model.

F.8. Model Distillation Results

Figure 17 illustrates the performance of SAM 3D with and without distillation, plotted against the number of flow matching steps for the Geometry model. Specifically, using 1-step and 4-step methods yields a 38× and 10× inference speed improvement, respectively, compared to 25 steps with flow matching (w/ CFG). In flow matching mode, CFG is applied during the first half of the steps, resulting in a total NFE that

[1]While some of this may be from human error, we attribute most of this disagreement to random chance when the MITL suite returns multiple similar meshes, or when there is no clear winner due to meshes being better or worse in different parts.

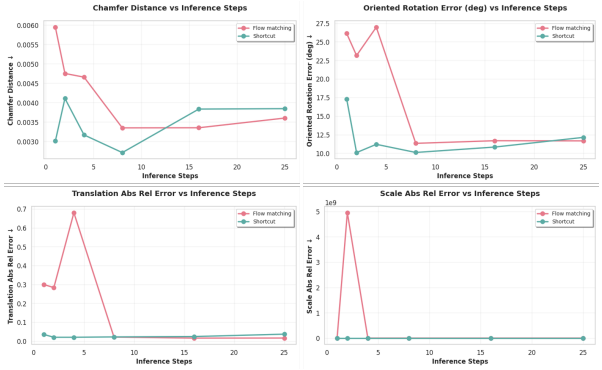

Figure 17. **Model distillation.** Geometry model shortcut versus flow matching. As shown in the plot, flow matching distillation enables the model to perform significantly better during the early iterations, nearly on par with 25 steps performance.

is 1.5 times the number of steps. In contrast, shortcut mode achieves an NFE equal to the number of steps, as CFG is distilled directly into the model. For the Texture & Refinement model, we opted not to apply distillation for the final release, as the final model already performs well with fewer steps out of the box. This is because the overall geometry is primarily determined by the Geometry model's voxel output, and increasing the number of steps does not significantly alter the geometry. However, as illustrated in Figure 18, where we present a visualization of the model outputs using different numbers of inference steps with shortcut mode enabled for both the Geometry model and the Texture & Refinement model, shortcut model distillation improves texture quality when using fewer steps.

F.9. Input Mask Robustness

SAM 3D assumes a segmentation mask as input to designate to the model which object within a scene should be reconstructed, as well as isolate the object's pixels for visual feature extraction. Accurate segmentation by hand can be time-consuming, but the ready availability of segmentation models like the SAM series [8, 44, 77] offer a fast, promptable solution. On the other hand, automatic methods are not perfect, and as a preceding step to SAM 3D, there is risk of compounding errors. Yet, we observe empirically that SAM 3D has fair robustness to errors in mask precision (see Figure 19), demonstrating the ability to recognize the intended target object and reconstruct it. We also note that as an input to the model, SAM 3D is agnostic to the source of mask, and thus is able to take advantage of ground truth masks or advancements in segmentation by future models.

G. Potential Negative Societal Impacts

SAM 3D represents a major step toward real-world 3D perception. Such capabilities have the potential to be transfor-

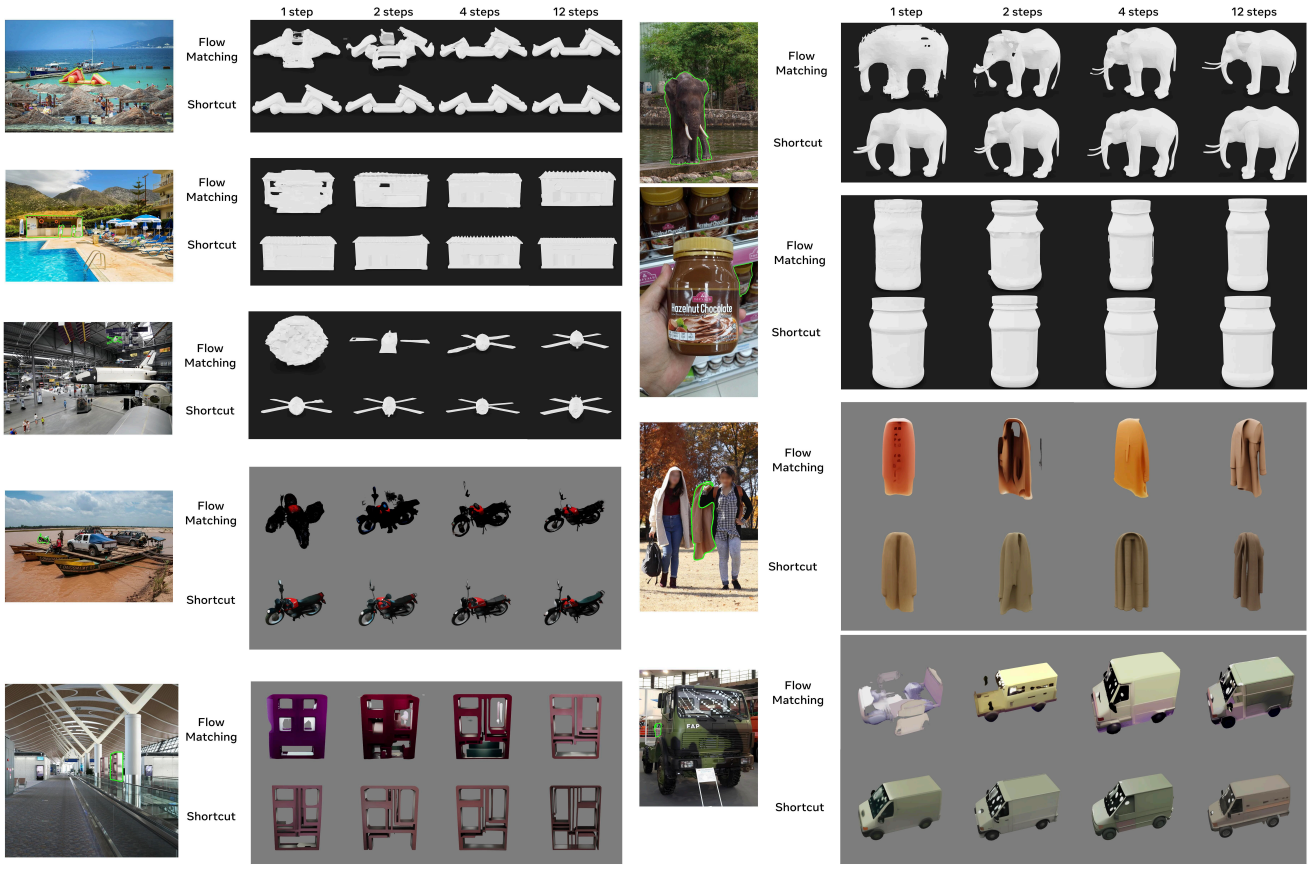

Figure 18. **Qualitative examples after distillation.** Visualization using difference number of steps in flow matching mode and shortcut mode. The black background displays the mesh rendering without texture, while the grey background shows the rendering of the Gaussian splattings.

Figure 19. **SAM 3D robustness to input mask precision. Top**: Over-segmentation; **Bottom**: Under-segmentation.

mative for many downstream applications, such as robotics, AR/VR, interactive media, and others. Many of these have the potential for exciting positive applications but as with many emerging technologies, there may also be risks of negative societal impacts as well. For example, robotics have many useful applications, such as household cleaning or assisting the elderly, where 3D perceptual capabilities will play a key role, but they can also be deployed for harmful use cases as well, including in military contexts when deployed irresponsibly. To foster open development on top of SAM 3D while guarding against more harmful applications, we release SAM 3D under a bespoke license which prohibits military use.

H. Limitations

There are limits to our model's resolution based on the architectural hyperparameters we used. The geometry model uses a coarse shape resolution of $O \in \mathbb{R}^{64^3}$; we trained multiple 3D Gaussian splat decoders, with up to 32 splats per occupied voxel. This is sufficient for many types of objects, but for more complex shapes, thin structures, or where human perception is especially attuned, these limits to resolution can lead to noticeable distortions or loss of details. For example, as part of a whole human body, the number of voxels/splats our chosen resolution is able to devote to hands or faces is inherently limited by the overall body's scale, and due to human visual system's acuity to such features, this can lead to perceptible artifacts to these body parts. By contrast, when focused on just a single hand or the head,

the higher relative resolution available means SAM 3D can reconstruct these significantly better. For these kinds of objects and others, a natural next step for SAM 3D would be to increase the output resolution via architectural changes, a superresolution model, parts-based generation, or switching to an implicit 3D representation.

Object layouts are another area where improvements can be made. SAM 3D predicts objects one at a time, and isn't trained to reason about physical interactions, such as contact, physical stability, interpenetration, or co-alignment (*e.g.* on the same ground plane). Multi-object prediction combined with appropriate losses would allow joint reasoning about multiple objects in a scene. Similarly, while SAM 3D's single-object inference is embarrassingly parallel across multiple GPUs, scene-based inference may also present opportunities to more efficiently accelerate the reconstruction of entire scenes. Finally, SAM 3D's texture predictions are made without knowledge of the predicted object's pose; as a result, for objects with rotational symmetry, it can occasionally predict textures that effectively rotate the object to the incorrect orientation.

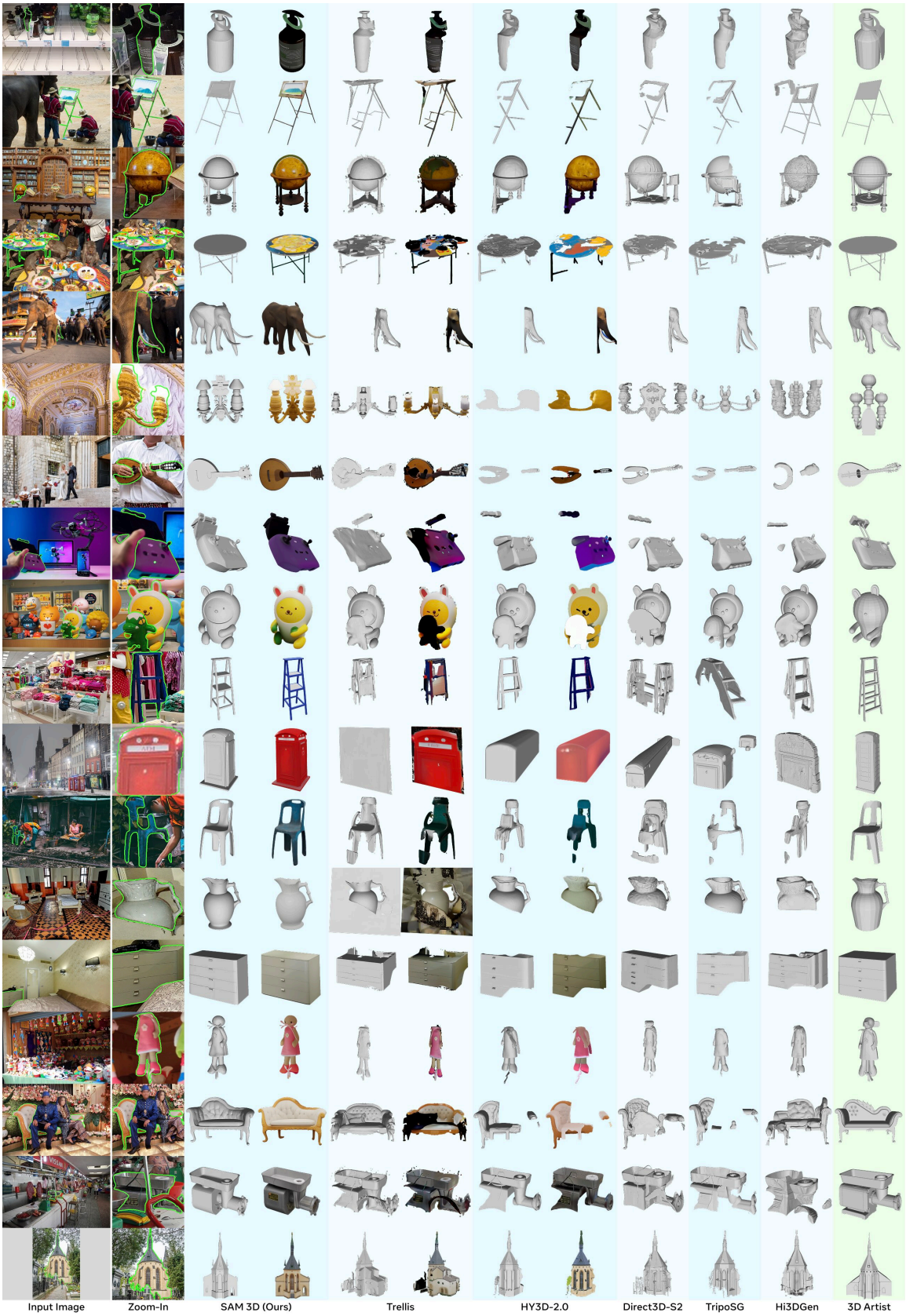

Figure 20. **Additional qualitative shape and texture results of our model on the SA-3DAO eval set.** For models that include texture, we show the untextured mesh (left) and textured mesh (right) separately.

Figure 21. **Qualitative examples for scene reconstruction.** Showing examples of SAM 3D and alternative scene reconstruction methods.