Which Reconstruction Model Should a Robot Use? Routing Image-to-3D Models for Cost-Aware Robotic Manipulation

Akash Anand Aditya Agarwal Leslie Pack Kaelbling
Massachusetts Institute of Technology
{akash10, adityaag, lpk}@mit.edu

Abstract

Robotic manipulation tasks require 3D mesh reconstructions of varying quality: dexterous manipulation demands fine-grained surface detail, while collision-free planning tolerates coarser representations. Multiple reconstruction methods offer different cost-quality tradeoffs, from Image-to-3D models—whose output quality depends heavily on the input viewpoint—to view-invariant methods such as structured light scanning. Querying all models is computationally prohibitive, motivating per-input model selection. We propose SCOUT, a novel routing framework that decouples reconstruction scores into two components: (1) the relative performance of viewpoint-dependent models, captured by a learned probability distribution $\hat{\mathbf{p}}$, and (2) the overall image difficulty, captured by a scalar partition function estimate $\hat{z}$. As the learned network operates only over the viewpoint-dependent models, view-invariant pipelines can be added, removed, or reconfigured without retraining. SCOUT also supports arbitrary cost constraints at inference time, accommodating the multi-dimensional cost constraints common in robotics. We evaluate on the Google Scanned Objects, BigBIRD, and YCB datasets under multiple mesh quality metrics, demonstrating consistent improvements over routing baselines adapted from the LLM literature across various cost constraints. We further validate the framework through robotic grasping and dexterous manipulation experiments. We release the code and additional results on our website.

1. Introduction

Single-image 3D reconstruction has advanced rapidly in recent years. Early optimization-based approaches demonstrated that strong generative priors [19, 34] could lift a single image into 3D, but required costly per-object optimization [28, 29, 33, 40] taking minutes to hours, limiting their use in real-time robotics applications. Recent Image-to-3D models remove this bottleneck, producing mesh reconstructions in seconds without per-object optimization. These models span diverse architectures—feed-forward prediction [9, 38], multi-view diffusion followed by reconstruction [17, 47], and iterative generation in learned 3D latent spaces [36, 45]—and offer complementary strengths: some models prioritize inference speed while others emphasize geometric fidelity or texture quality, and their relative performance varies across object categories and input viewpoints. As a result, no single method dominates.

This is relevant for robotic manipulation, where a robot often has access to only a single initial view of an object [25, 32], and acquiring additional viewpoints is time-consuming and not always feasible [22, 24]. Several works use Image-to-3D models for perception in downstream manipulation [1, 11, 12, 14, 25]. However, these works fix the choice of Image-to-3D model, forgoing potential gains from selecting the best model per input. Moreover, whether a given reconstruction is sufficient depends on the task: dexterous manipulation demands fine-grained surface detail, whereas collision-free motion planning tolerates coarser representations. Beyond Image-to-3D models, methods such as structured light scanning provide reconstructions whose quality depends on the object and hardware configuration rather than on the robot's initial viewpoint, but they require additional sensing time. We refer to such methods as *view-invariant*, in contrast to the *viewpoint-dependent* Image-to-3D models above. The choice of reconstruction method therefore depends on the input viewpoint, task requirements, and available budget.

We address this problem via model routing: selecting a reconstruction model for a given input prior to inference. The routing decision balances reconstruction accuracy against user-specified costs—including latency, memory, and monetary expenses. While model routing has emerged as an active research area for LLMs [3, 10, 16, 21, 26], it has not been explored for 3D reconstruction. The two settings differ in important ways: the costs of reconstruction models are largely input-independent, robotics applications require multi-dimensional cost coefficient vectors rather than a single scalar tradeoff, and the model pool in-

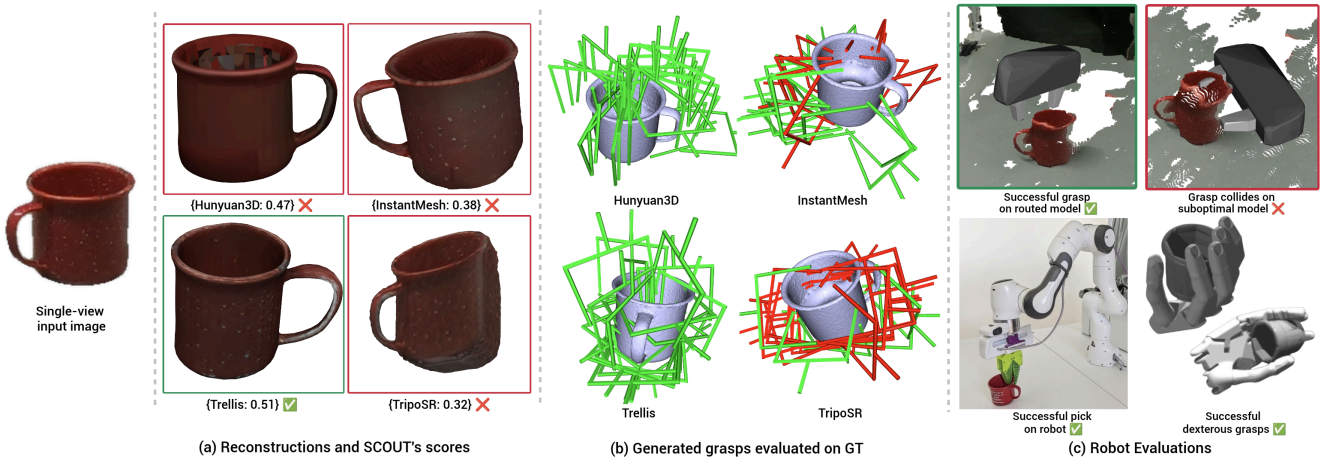


Figure 1. Overview of SCOUT. Given an input image, (a) SCOUT routes to the most suitable reconstruction model and generates the selected 3D reconstruction (compared against other candidate models). (b) Grasp proposals generated on the reconstructed mesh, evaluated on the ground-truth mesh; colliding grasps are shown in red. (c) Utility of SCOUT's reconstruction-aware routing in downstream robust robot grasping and dexterous manipulation.

cludes both viewpoint-dependent methods (*e.g.*, Image-to-3D models) and view-invariant methods that can be reconfigured post-deployment, such as by changing the number of views captured.

We propose **SCOUT** (**S**core-**C**onditioned **O**ptimal **U**tility **T**argeting), a routing framework that decouples reconstruction scores into two components: (1) the relative performance of viewpoint-dependent models, captured by a learned probability distribution, and (2) the overall difficulty of the input image, captured by a scalar partition function estimate. This decoupling stabilizes training and provides a key scalability advantage: the learned network predicts only over the viewpoint-dependent models, so its size and training procedure are entirely independent of the view-invariant methods. SCOUT also supports arbitrary cost coefficient vectors at inference time without retraining.

We evaluate SCOUT on the Google Scanned Objects [6], BigBIRD [35], and YCB [2] datasets under multiple mesh quality metrics, including Density-aware Chamfer Distance (DCD) [43], IoU, and geometric metrics from Eval3D [7]. SCOUT consistently outperforms routing baselines adapted from the LLM literature across diverse cost coefficient vector subspaces, and we validate its practical utility through robotic manipulation experiments, including collision-free grasp proposal evaluation, dexterous manipulation in simulation, and real-world pick-and-place on a Franka Panda robot.

The main contributions of this work are: (1) the first formulation of model routing for 3D reconstruction, accommodating viewpoint-dependent and view-invariant methods under arbitrary cost constraints; (2) a theoretically justified partition function proxy that recovers absolute scores from the learned relative distribution with provably optimal weighting; and (3) a decoupling procedure separating image difficulty from relative model performance, enabling view-invariant methods to be added without retraining.

2. Related Work

Mesh reconstruction models. Early single-image 3D reconstruction methods relied on per-object optimization, using score distillation sampling [28, 40] or novel view synthesis models [19, 34] to iteratively optimize a 3D representation from a single image [29, 33]. Subsequent methods replaced per-object optimization with fast inference [9, 17, 18]. Among recent and widely used Image-to-3D models, TripoSR [38] prioritizes low-latency inference via a feed-forward transformer, Hunyuan3D [36] emphasizes high-quality texture generation through a two-stage multi-view diffusion and reconstruction pipeline, InstantMesh [47] focuses on geometrically plausible reconstructions with accurate depths and surface normals, and TRELLIS [45] employs a unified structured latent representation that captures intricate geometric detail. However, these models exhibit complementary failure modes, with performance varying across object categories and viewpoints. The diversity of available reconstruction pipelines motivates the need for a principled method to select the most appropriate reconstruction model given a specific input and user-defined cost constraints.

Routing models. Querying all models for each input is computationally intractable, motivating a routing network to select a single reconstruction model per input. To the best of our knowledge, routing has not been explored for 3D reconstruction, though it has emerged as an active research area for LLMs. ZOOTER [21] trains a router via distillation from a reward model, and RouterDC [3] learns per-model embeddings using contrastive losses. While both achieve

strong routing performance, neither supports adjusting the cost-quality tradeoff at inference time without retraining.

Recognizing this limitation, RouterBench [10] introduces baselines and metrics for evaluating routing models that can operate across varying cost-quality tradeoffs without retraining. Li [16] analyzes the proposed baselines and additional architectures, finding that simple models such as kNNs and linear regressors consistently outperform more complex neural approaches.

3. Methods

In this section, we first formalize the model routing problem for the 3D reconstruction domain and then introduce SCOUT, a routing framework that decouples reconstruction scores into the relative performance of viewpoint-dependent models conditioned on the input image and a scalar shift term capturing the overall difficulty of the input image.

3.1. Problem formulation

Let $\mathcal{M} = \{m_1, \cdots, m_k\}$ denote a set of k candidate 3D reconstruction models, each producing a mesh from an input image of the current robot viewpoint, x. We seek a router that selects the optimal model m^* for input x, determined by: a score function $s(x, m)$ that measures reconstruction quality (higher is better), and an image-independent cost function $c(m)$ that captures a user-specified penalization of each model. Given a training dataset $\mathcal{D}_{train} = \{(x^{(1)}, \mathbf{s}^{(1)}), \cdots, (x^{(n)}, \mathbf{s}^{(n)})\}$, where $\mathbf{s}^{(i)} \in \mathbb{R}^k$ denotes the vector of reconstruction scores $s(x^{(i)}, m_j)$ for $j \in [k]$, we aim to learn a router h that takes an input image x and an arbitrary cost coefficient vector $[c(m_1), \cdots, c(m_k)] \equiv \mathbf{c} \in \mathbb{R}^k$, and outputs a model selection $\hat{m}$ to maximize $s(x, \hat{m}) - c(\hat{m})$.

In the LLM setting, prior work [13, 16] defines the most general routing problem as maximizing the following objective for a user-defined scalar λ:

$$m^* = \arg\max_{m \in \mathcal{M}} s(x, m) - \lambda \cdot c(x, m). \tag{1}$$

Our formulation differs in three aspects: the dependence of c on m rather than on both x and m, the use of an arbitrary cost coefficient vector $\mathbf{c} \in \mathbb{R}^k$ rather than a scalar-weighted cost $\lambda \cdot \mathbf{c}_{\text{fixed}}$, and the composition of the model pool $\mathcal{M}$, which in our setting includes both viewpoint-dependent Image-to-3D models and view-invariant methods. We expand on these differences in turn.

First, in the LLM setting, inference cost is a function of the number of input and output tokens [27], both of which depend on the input prompt. Therefore, the cost function depends on both the input prompt and the queried model. In contrast, 3D reconstruction models often employ architectures with fixed computational graphs (*e.g.*, pure feed-forward [38] or fixed-step diffusion pipelines [47]), resulting in a constant inference cost. Even when this does not hold exactly, the approximation $c(x, m) \approx c(m)$ remains valid in practice and is adopted in several LLM routing works [10, 15].

Second, as the LLM routing problem originated from selecting between paid API endpoints, cost is expressed in dollars per API call. In contrast, routing between 3D reconstruction models for robotics applications requires a more task-specific cost structure. The cost may simultaneously reflect both memory footprint and inference latency, spanning two dimensions rather than the one-dimensional family $\{\lambda \cdot \mathbf{c}_{\text{fixed}} : \lambda \in \mathbb{R}\}$ traced by sweeping a single scalar λ over a fixed cost vector. The cost structure can be considerably more complex in practice: a robot may assign a cost of ∞ to models exceeding its memory budget, yielding a binary feasibility constraint. We therefore argue that a routing framework must support an arbitrary cost coefficient vector $\mathbf{c} \in \mathbb{R}^k$, rather than a single scalar tradeoff parameter.

Third, unlike LLM routing, which considers only generative models, 3D reconstruction includes two fundamentally different pipeline classes. Viewpoint-dependent models, such as Image-to-3D models, produce scores that depend on both the input image and the model—a frontal (face-on) view can yield far worse reconstructions than a corner view. View-invariant methods, such as laser scanning and structured light scanning, capture their own viewpoints, so their reconstruction quality is largely independent of the robot's initial view. The quality instead depends on object-level properties such as texture and scanner configurations (*e.g.*, the number of captured views and sensor resolution).

This distinction has two implications for routing. First, because view-invariant methods are not conditioned on the input view, including them in a joint learned model alongside viewpoint-dependent methods is both unnecessary and harmful. It is unnecessary because their scores can be estimated without a learned image-conditioned network—for example, via texture analysis, object priors, or as a function of the scanner configurations. It is harmful because the training objective for viewpoint-dependent models benefits from reasoning about *relative* performance across models for a given input (Sec. 3.2), and mixing in scores that do not vary with the input view corrupts this relative signal. Second, the number of view-invariant configurations can be large—different scanners, view counts, and hardware setups each constitute a distinct option with its own cost-quality tradeoff—and a robot's available hardware may change between deployments. Coupling these into the learned network would require retraining whenever a new configuration is added and degrades performance as the number of configurations grows, as shown in Fig. 2. Handling them separately avoids both problems.

In our experiments, we approximate view-invariant methods with fixed scores $s(m)$ that depend only on the

method. This approximation is reasonable for our datasets with diffuse textures [6], but in general $s(m)$ can be replaced with many estimation techniques, such as those listed above.

3.2. SCOUT

SCOUT is designed for two properties: (1) *scalability* — view-invariant methods can be added or reconfigured without retraining; and (2) *flexibility* — routing under arbitrary $\mathbf{c} \in \mathbb{R}^k$. The framework consists of three stages: preprocessing, score decoupling, and utility optimization.

Preprocessing. Let $\mathbf{s}^{(i)} = (s_1^{(i)}, \cdots, s_k^{(i)})$ denote the vector of reconstruction scores for all k models on input $x^{(i)}$. We partition this vector into scores for the k_1 viewpoint-dependent models, and scores for the k_2 view-invariant models, such that $\mathbf{s}^{(i)} \equiv \left[\mathbf{s}_{\text{dep}}^{(i)}, \mathbf{s}_{\text{inv}}^{(i)}\right]$ with $\mathbf{s}_{\text{dep}}^{(i)} \in \mathbb{R}^{k_1}$, $\mathbf{s}_{\text{inv}}^{(i)} \in \mathbb{R}^{k_2}$, and $k_1 + k_2 = k$.

Following ZOOTER [21] and motivated by the variance in reconstruction quality across object categories, we apply tag-based score smoothing to $\mathbf{s}_{\text{dep}}^{(i)}$ by grouping objects into semantic categories and computing category-averaged scores. We denote this grouping as a tagger $\mathcal{T} : \mathcal{X} \to \text{tag}$, and define the tag-averaged scores $\mathbf{s}_{\mathcal{T}(x^{(i)})}$ as the mean of $\mathbf{s}_{\text{dep}}^{(i)}$ over all training examples assigned to the same tag as $x^{(i)}$. The smoothed scores are then given by:

$$\tilde{\mathbf{s}}_{\text{dep}}^{(i)} = \beta \mathbf{s}_{\text{dep}}^{(i)} + (1 - \beta)\mathbf{s}_{\mathcal{T}(x^{(i)})}, \tag{2}$$

where $\beta \in [0, 1]$ is a hyperparameter controlling the degree of smoothing. Smoothing is applied only when learning relative model performance, as described below; the final recovered scores target the original unsmoothed values $\mathbf{s}_{\text{dep}}^{(i)}$, which are needed for comparison with $\mathbf{s}_{\text{inv}}^{(i)}$.

Learned models. The smoothed scores $\tilde{\mathbf{s}}_{\text{dep}}^{(i)}$ can be predicted using several possible training objectives, including MSE directly on the smoothed scores, or KL divergence on the softmax distribution of the smoothed scores. We adopt the KL divergence formulation for the following reason: an easy input image (*e.g.*, a corner view) may yield uniformly high scores across all viewpoint-dependent models, while a hard image (*e.g.*, frontal) may yield uniformly low scores. A direct regression objective such as MSE treats these absolute score values as the learning target, whereas KL divergence operates on the relative performance of the viewpoint-dependent models by normalizing out image difficulty. This is precisely why decoupling is necessary: KL divergence factors out image difficulty only when all models in the softmax share the same difficulty-induced offset, which holds for viewpoint-dependent models but not for view-invariant methods. For each input image, we convert the smoothed scores $\tilde{\mathbf{s}}_{\text{dep}}^{(i)}$ into a probability distribution via the softmax with temperature T:

$$p_j^{(i)} = \frac{e^{\tilde{s}_{\text{dep},j}^{(i)}/T}}{\sum_{m=1}^{k_1} e^{\tilde{s}_{\text{dep},m}^{(i)}/T}}, \tag{3}$$

and train a neural network to map images x into predictions $\hat{\mathbf{p}} \in \mathbb{R}^{k_1}$ by minimizing the KL divergence.

Our approach decouples the predicted scores into two interpretable components: the relative performance of the viewpoint-dependent models, captured by $\hat{\mathbf{p}}$, and the overall difficulty of the input image, captured by a scalar estimate $\hat{z}$. Importantly, $\hat{z}$ applies a uniform offset to all viewpoint-dependent scores, preserving their relative utilities while enabling direct comparison with view-invariant scores $\mathbf{s}_{\text{inv}}$.

Recovering the unsmoothed scores from $\hat{\mathbf{p}}$ requires estimating a partition function. We define $z^{(i)} = \sum_{m=1}^{k_1} e^{s_{\text{dep},m}^{(i)}/T}$ over the original unsmoothed scores, so that $T \cdot \ln(\hat{p}_j^{(i)} \cdot z^{(i)})$ approximates $s_{\text{dep},j}^{(i)}$. If only viewpoint-dependent models were present, $T \cdot \ln(\hat{\mathbf{p}})$ could be used directly for utility optimization, as all scores would be offset by the same constant $T \cdot \ln(z^{(i)})$. However, since the viewpoint-dependent scores must be compared against the view-invariant scores $\mathbf{s}_{\text{inv}}$, an estimate $\hat{z}$ is required to recover absolute score values.

We learn $\hat{z}$ via a second supervised model that takes x as input. Because the scores are noisy, we use per-model proxies

$$\tilde{z}_j^{(i)} = \frac{e^{s_{\text{dep},j}^{(i)}/T}}{\hat{p}_j^{(i)}} \tag{4}$$

rather than using $z^{(i)}$ directly. Note that unsmoothed scores appear in the proxy: smoothing blends per-image scores toward category averages, suppressing the per-image variation that z must capture (see Sec. 8). This yields k_1 proxies for $z^{(i)}$, which must be combined into a single estimate since all viewpoint-dependent scores are offset by the same constant $T \cdot \ln(z)$. As derived in Sec. 7, the optimal combination can be approximated by a weighted average:

$$\tilde{z}^{(i)} = \sum_{j=1}^{k_1} w_j \tilde{z}_j^{(i)}, w_j \propto \frac{1}{S_j^2(1 - R_j^2)\mathbb{E}_i\left[\frac{z^4}{e^{2s_{\text{dep},j}/T}}\right]} \tag{5}$$

where R_j^2 is the coefficient of determination of the predicted probabilities $\hat{p}_j$ relative to the true values p_j for model m_j and $S_j^2 = \text{Var}[\hat{p}_j]$. The resulting $\tilde{z}^{(i)}$ values serve as regression targets for the second supervised model. We compare this weighting against three alternatives: the ground-truth $z^{(i)}$, equal weighting, and one-hot weighting ($\tilde{z}^{(i)} = \tilde{z}_{j^*}^{(i)}$ where $j^* = \arg\max_j w_j$), and show improved performance using a one-sided t-test (see Sec. 11.1).

Utility optimization. Given the predicted probability distribution $\hat{\mathbf{p}}$ and the estimated partition function $\hat{z}$,

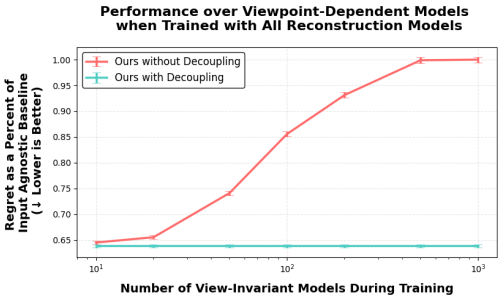


Figure 2. Effect of the number of view-invariant methods on routing performance when evaluated on viewpoint-dependent methods. With decoupling, regret over viewpoint-dependent models remains constant regardless of how many view-invariant methods are included during training. Without decoupling, regret increases as the number of view-invariant methods grows.

the optimal reconstruction method is selected as $\hat{m} = \arg\max_{m_j \in \mathcal{M}} \hat{s}(x, m_j) - \mathbf{c}[j]$, where the estimated score is defined as:

$$\hat{s}(x, m_j) = \begin{cases} T \cdot \ln(\hat{p}_j \cdot \hat{z}), & \text{if } j \leq k_1, \\ s_{\text{inv}}(m_j), & \text{if } k_1 < j \leq k. \end{cases} \tag{6}$$

For view-invariant methods, $s_{\text{inv}}(m_j)$ may depend on object-level properties, such as texture or reflectance, but it does not depend on the viewpoint from which the robot initially observes the object. Because the learned network only predicts $\hat{\mathbf{p}}$ over the k_1 viewpoint-dependent models, new view-invariant configurations can be added, removed, or reconfigured at inference time without retraining. In contrast, applying the softmax over the full score vector $\mathbf{s}^{(i)}$, including view-invariant scores, would couple the learned distribution to k_2, resulting in degraded performance as the number of view-invariant methods increases, as demonstrated in Fig. 2.

4. Experiments

In this section, we describe the experimental setup, evaluate SCOUT on three benchmark datasets, and present real-world robotic manipulation results.

4.1. Experimental setup

3D reconstruction models. We evaluate SCOUT using four Image-to-3D models: Hunyuan3D [36], InstantMesh [47], TripoSR [38], and TRELLIS [45]. These models were selected to ensure architectural diversity, giving rise to complementary biases across object categories and input viewpoints. In addition, we include a view-invariant reconstruction option, approximated as a fixed score since our datasets contain objects with diffuse textures [6], as discussed in Sec. 3.1.

Datasets and reconstruction pipeline. Training the router requires a dataset of paired inputs and reconstruction scores. We begin with a collection of ground-truth 3D meshes and corresponding images, obtained either through rendering or real-world capture. For each dataset image, we run all models and score each reconstruction against the ground-truth mesh to produce $\mathbf{s}^{(i)} \in \mathbb{R}^k$.

We evaluate SCOUT on standard datasets: the Google Scanned Objects (GSO) dataset [6] and the YCB dataset [2] (supplemented with real images from BigBIRD [35]). For the GSO dataset, which contains 1,030 objects, we render 15 images per object and 15,450 images in total. All viewpoints are nondegenerate, as the azimuth and elevation angles are chosen to avoid direct face-on views of the object.

For the YCB dataset, we select 54 objects that have corresponding real images from the BigBIRD dataset. For each object, we select 13 images spanning 5 elevation angles spaced from $0°$ to $90°$; the top view ($90°$) is captured at a single azimuth angle, while the remaining elevations are captured at azimuth angles of $0°$, $45°$, and $90°$. Unlike the GSO dataset, many of these viewpoints are degenerate, capturing the object face-on with limited depth information, making this a more challenging evaluation setting. For each viewpoint, we collect a real image from the BigBIRD dataset, a flash-style lighting render, and a surround-style lighting render. This yields 39 images per object and 2,106 images in total. The combination of challenging viewpoints, lighting variation, and limited data makes this setting substantially different from the GSO dataset.

For each image, we reconstruct a mesh with each model, normalize it to a unit sphere, register via FoundationPose [42] followed by DDM [31], and then evaluate using a reconstruction quality metric. Our primary metric is Density-aware Chamfer Distance (DCD) [43], which we negate so that higher scores indicate better reconstructions, with additional results reported for Chamfer Distance, IoU, and geometric metrics from Eval3D [7].

Baselines. To the best of our knowledge, no prior routing methods exist for the 3D reconstruction domain; therefore, we adapt and evaluate several routing architectures from the LLM literature. Specifically, we compare against: (i) ZOOTER [21], which converts the full score vector—including view-invariant scores—to a probability distribution and trains a supervised router on the resulting targets; (ii) RouterDC [3], which learns a latent embedding space for each model as well as a query-dependent image representation using contrastive losses; (iii) an MLP baseline from RouterBench [10]; (iv) a matrix factorization (MF) method based on RouteLLM [26]; and (v) kNN and linear regression (LR) models, which have been shown to frequently outperform more complex neural architectures in the LLM routing setting [16]. The most similar of these to our method is ZOOTER, with two key differences: SCOUT decouples viewpoint-dependent and view-invariant scores, and predicts scores directly rather than selecting the

highest-ranked model, enabling support for arbitrary cost coefficient vectors and additional view-invariant methods without retraining.

We report performance relative to an input-agnostic baseline, which computes the average reconstruction score for each model across all training examples, yielding a fixed score estimate $\bar{s}(m_j) = \frac{1}{n}\sum_{i=1}^{n} s(x^{(i)}, m_j)$ per model, and selects $\arg\max_{m_j} \bar{s}(m_j) - \mathbf{c}[j]$.

For the evaluation metrics prescribed by RouterBench, we additionally include the Zero router baseline [10], which randomly samples routers at a given total cost allocation to trace out the non-decreasing convex hull (shown in Fig. 3). The Zero router is not included in other tables, as those evaluate average regret over cost coefficient vector subspaces rather than utility at a given total cost allocation.

Image representation. The router takes as input concatenated features from CLIP ViT-B/32 [30], ResNet-50 [8], ViT-B/16 [5], and ConvNeXt-B [20].

Evaluations. Since reconstruction quality is continuous and metric-dependent, with no notion of a perfectly correct output, we evaluate routing performance in terms of average utility or average regret over various cost subspaces.

Similar to splitting $\mathbf{s}^{(i)}$ (Section 3.2), we split the cost coefficient vector as $\mathbf{c} \equiv [\mathbf{c}_{\text{dep}}, \mathbf{c}_{\text{inv}}]$ for ease of notation. The first subspace is the single cost coefficient vector $\{\mathbf{c}_{\text{dep}} = \mathbf{0} \cap \mathbf{c}_{\text{inv}} = \infty\} \equiv \mathcal{C}_0$, forcing selection among viewpoint-dependent models only. This also enables comparison with ZOOTER and RouterDC, which do not support varying costs at inference time. The second subspace consists of the one-dimensional cost families $\lambda \cdot \mathbf{c}_{\text{latency}}$, $\lambda \cdot \mathbf{c}_{\text{memory}}$, and $\lambda \cdot (\mathbf{c}_{\text{latency}} \odot \mathbf{c}_{\text{memory}})$, which penalize inference latency, memory, and memory-time occupancy (GB·s), respectively, each swept over a range of λ values. For these cost families, we report additional metrics including the cost-deferral curve and Average Improvement in Quality (AIQ) [10]. As mesh reconstruction scores are continuous rather than discrete, we normalize the y-axis by the maximum achievable utility under an oracle router with no cost penalization, rather than reporting accuracy.

The third subspace is the full "interesting" cost subspace $\mathcal{C}$, defined to capture cost coefficient vectors that place both the viewpoint-dependent and view-invariant models in comparable utility ranges, thereby ensuring non-trivial routing decisions. For each method m_j, we compute the 75th percentile $P_{75,j}$ and interquartile range IQR_j of its score distribution. We then set the cost range for c_j to $[P_{75,j} - \min_j(\text{IQR}_j),\ P_{75,j}]$, where $\min_j(\text{IQR}_j)$ ensures a uniform range width across all models.

4.2. GSO results

Table 1 shows the regret of each method across different cost subspaces on novel objects using DCD. SCOUT achieves statistically significant lower regret than all baselines across all cost subspaces. For subspace $\mathcal{C}_0$, ZOOTER performs most similarly to SCOUT. The benefit of decoupling becomes more pronounced in the broader subspace $\mathcal{C}$, where routing decisions are more sensitive to precise score predictions, as the routing decisions are all nontrivial.

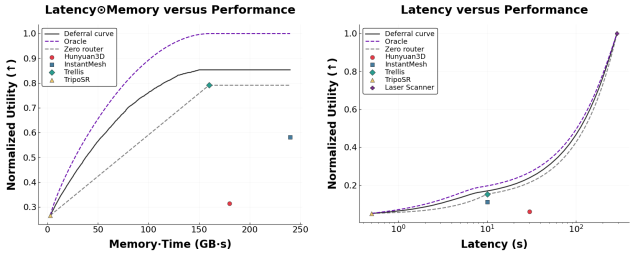
Table 1. Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset.

Method	$\{\mathbf{c} \in \mathcal{C}_0\}$	$\{\mathbf{c} \in \mathcal{C}\}$	$\{\mathbf{c} \in \mathcal{C} \cap \mathbf{c}_{\text{inv}} = \infty\}$
ZOOTER	0.6489 ± 0.0038	N/A	N/A
RouterDC	0.8013 ± 0.0050	N/A	N/A
MLP	0.6716 ± 0.0038	0.4653 ± 0.0026	0.5004 ± 0.0021
MF	0.7557 ± 0.0066	0.5163 ± 0.0036	0.5814 ± 0.0040
kNN	0.6489 ± 0.0033	0.4630 ± 0.0022	0.4992 ± 0.0019
LR	0.6497 ± 0.0033	0.4519 ± 0.0021	0.5047 ± 0.0019
Ours (no decoupling)	0.6489 ± 0.0038	0.4511 ± 0.0021	0.4882 ± 0.0018
Ours	$\mathbf{0.6386 \pm 0.0033}$	$\mathbf{0.4319 \pm 0.0021}$	$\mathbf{0.4831 \pm 0.0019}$
Input-agnostic	1.0000	1.0000	1.0000
Always HY3D	3.0730 ± 0.0091	1.9693 ± 0.0054	1.2863 ± 0.0063
Always IM	1.9272 ± 0.0049	2.5788 ± 0.0052	1.8690 ± 0.0040
Always TRL	1.0000	2.4762 ± 0.0056	1.7709 ± 0.0053
Always TSR	3.1118 ± 0.0079	2.5109 ± 0.0057	1.8041 ± 0.0051

For the one-dimensional cost families, we evaluate using the standard metrics from RouterBench. By sweeping λ, we obtain different achieved utilities at different total cost allocations, from which we can construct a Pareto frontier $(R_h(c))$ as shown in Fig. 3. The AIQ, defined as:

$$\text{AIQ}(h) = \frac{1}{c_{max} - c_{min}} \int_{c_{min}}^{c_{max}} R_h \, \mathrm{d}c \tag{7}$$

is shown in Tab. 2. ZOOTER and RouterDC are excluded from this evaluation as they do not support varying λ without retraining. The deferral curves show that the presence of a high-quality but high-latency method (such as structured light scanning) reduces the margin for possible improvement, as the difference in AIQ between the Zero router and an oracle router is very small. SCOUT achieves the highest AIQ for memory and latency⊙memory, and is competitive on latency, where MLP performs equally well. The magnitude of improvement over baselines is consistent with the inter-method differences reported in RouterBench [10], where gains between routing methods are similarly modest.



(a) Latency⊙Memory deferral curve. (b) Latency deferral curve.

Figure 3. Deferral curves for the latency⊙memory (a) and latency (b) cost coefficient vectors, showing the normalized utility achieved by each method as a function of the allocated cost.

Table 2. AIQ (↑) for one-dimensional cost families on the GSO dataset, evaluated on novel objects.

Method	Latency⊙Memory	Latency	Memory
MLP	0.7365 ± 0.0008	$\mathbf{0.5815 \pm 0.0003}$	0.7911 ± 0.0007
MF	0.7233 ± 0.0011	0.5795 ± 0.0003	0.7736 ± 0.0011
kNN	0.7382 ± 0.0007	0.5798 ± 0.0002	0.7942 ± 0.0006
LR	0.7364 ± 0.0007	0.5787 ± 0.0002	0.7913 ± 0.0006
Ours (no dc)	0.7408 ± 0.0007	0.5796 ± 0.0003	0.7958 ± 0.0006
Ours	$\mathbf{0.7416 \pm 0.0007}$	0.5814 ± 0.0002	$\mathbf{0.7976 \pm 0.0006}$
Zero router	0.6248 ± 0.0006	0.5628 ± 0.0002	0.6506 ± 0.0009
Oracle	0.8557 ± 0.0006	0.6066 ± 0.0002	0.9342 ± 0.0003

4.3. BigBIRD and YCB results

We now evaluate on BigBIRD + YCB across multiple scoring metrics. Varying the metric is important because different metrics induce different optimal routing policies: IoU is a volume-based metric that strongly penalizes models that fill in hollow geometry, whereas DCD measures surface-level point correspondences and penalizes local geometric inaccuracies regardless of volume. Since different robotics tasks may require different quality criteria, a routing framework must perform well across metrics. We report results for $\mathcal{C}_0$ in Tab. 3 and $\mathcal{C}$ in Tab. 4 for evaluation on novel objects. On average, models produce the highest regret on Eval3D metrics, as these scores rely on generative models [19, 48], introducing additional noise to an already challenging dataset. SCOUT is robust across metrics, from easier ones such as IoU (where all methods achieve relatively low regret) to noisier Eval3D metrics: it most frequently achieves the lowest regret and is the only method ranking in the top three across all metrics in Tabs. 3 and 4.

4.4. Robotics results

We evaluate whether SCOUT selects reconstruction models that improve downstream robotic manipulation. We test across three tasks: collision-free grasp proposal generation in simulation, dexterous manipulation in simulation, and real-world pick-and-place on a Franka Panda robot.

Grasp proposal evaluation. Following [1], we generate grasp proposals on each reconstructed mesh using a Robotiq two-fingered gripper and evaluate against the ground truth. We report two metrics: the grasp collision rate (fraction of proposals that result in collision with the ground-truth geometry; lower is better) and mesh-IoU between the reconstruction and ground truth (higher is better). Results across 10 YCB objects are shown in Tab. 5. We evaluate SCOUT under two settings: routing over novel views of known objects, and the harder problem of routing over both novel views and novel objects. SCOUT achieves the lowest mean collision rate and highest mean mesh-IoU under both evaluation settings. The per-object results confirm that no single reconstruction model dominates across objects, and that per-input model selection translates to measurable gains in downstream grasp quality.

Dexterous manipulation. We next evaluate whether reconstruction quality affects dexterous grasping success. Following [1], we test each reconstruction model in simulation across 5 objects and report the dexterous grasp success rate. Results are shown in Tab. 6. Dexterous manipulation places stricter demands on surface accuracy than two-fingered grasping, as contact placement and force closure depend on local surface geometry. This setting thus amplifies the performance differences between reconstruction models and further motivates per-input model selection.

Real-world pick and place. Finally, we validate the routing framework on a physical Franka Panda robot performing tabletop pick-and-place. For each of 5 objects, we reconstruct the object from a single input image using each Image-to-3D method, generate antipodal grasp proposals using DA2 [23], and randomly sample 5 proposals for execution. We report the pick-and-place success rate in Tab. 7. This experiment tests the full pipeline end-to-end: a single input image is routed to a reconstruction model, the resulting mesh is used for grasp planning, and the planned grasp is executed on the robot. The router's selections reflect the difficulty of the input viewpoint: for objects 1, 3, and 4, where the viewpoints are more challenging, SCOUT selects Hunyuan3D, which produces the highest-quality reconstructions under difficult conditions. For objects 2 and 5, where the viewpoints are more straightforward, the router selects TRELLIS, which suffices for these easier inputs and has lower latency.

5. Conclusion

We presented SCOUT, a routing framework that selects among 3D reconstruction models for a given input image under arbitrary cost-quality tradeoffs. The key idea is to decouple reconstruction scores into relative model performance and image difficulty, which keeps the learned network independent of the number of view-invariant methods and supports arbitrary cost coefficient vectors at inference time without retraining. SCOUT consistently outperforms routing baselines adapted from the LLM literature across three datasets, multiple quality metrics, and diverse cost coefficient vector subspaces, with the largest gains in the most challenging cost regions. Robotic manipulation experiments in simulation and the real world confirm routing improvements translate to downstream robotic performance.

Limitations & Future Work: Our evaluation uses four Image-to-3D (viewpoint-dependent) models; scaling to a larger model pool is a natural next step. Additionally, while our formulation supports view-invariant methods, our experiments approximate them with fixed scores rather than evaluating with real scanning hardware; validating with actual view-invariant pipelines is an important direction for future work. Integrating the router into a closed-loop manipulation system, where task feedback informs future routing decisions, is another promising direction.

Table 3. Regret (↓) as a proportion of the input-agnostic baseline regret for $\mathcal{C}_0$ across multiple quality metrics on BigBIRD + YCB. Ours (no dc) denotes our method without decoupling, equivalent to ZOOTER extended to support arbitrary cost vectors.

Method	DCD	Chamfer L2	Chamfer L1	IoU	MMD-EMD	Eval3D-geo	Eval3D-struct
MLP	1.1628 ± 0.0147	1.1677 ± 0.0252	1.0268 ± 0.0156	0.7135 ± 0.0100	0.9354 ± 0.0147	1.1520 ± 0.0431	1.2257 ± 0.0209
MF	1.3116 ± 0.0152	1.3869 ± 0.0247	1.2151 ± 0.0169	0.8972 ± 0.0172	1.3003 ± 0.0222	1.6351 ± 0.0791	1.3365 ± 0.0218
kNN	1.0385 ± 0.0118	0.9109 ± 0.0134	0.8818 ± 0.0093	0.7411 ± 0.0089	0.9604 ± 0.0158	0.9957 ± 0.0531	1.1563 ± 0.0158
LR	0.9983 ± 0.0120	0.9174 ± 0.0147	0.8506 ± 0.0105	$\mathbf{0.6421 \pm 0.0090}$	0.8554 ± 0.0139	$\mathbf{0.8246 \pm 0.0274}$	$\mathbf{1.0815 \pm 0.0159}$
Ours (no dc)	1.0171 ± 0.0129	0.8806 ± 0.0152	0.8395 ± 0.0121	0.6849 ± 0.0088	0.8447 ± 0.0143	0.8674 ± 0.0412	1.1197 ± 0.0167
Ours	$\mathbf{0.9767 \pm 0.0113}$	$\mathbf{0.8730 \pm 0.0146}$	$\mathbf{0.8222 \pm 0.0106}$	0.6609 ± 0.0091	$\mathbf{0.8136 \pm 0.0144}$	0.8588 ± 0.0413	1.1527 ± 0.0188
Input-agnostic	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4. Regret (↓) as a proportion of the input-agnostic baseline regret for $\mathbf{c} \in \mathcal{C}$ across multiple quality metrics on BigBIRD + YCB. Ours (no dc) denotes our method without decoupling.

Method	DCD	Chamfer L2	Chamfer L1	IoU	MMD-EMD	Eval3D-geo	Eval3D-struct
MLP	0.9944 ± 0.0165	1.1708 ± 0.0234	0.8983 ± 0.0149	0.7054 ± 0.0115	0.8153 ± 0.0128	3.1921 ± 0.1267	0.8975 ± 0.0117
MF	1.2518 ± 0.0158	2.7886 ± 0.0524	1.7604 ± 0.0281	0.8378 ± 0.0119	1.8527 ± 0.0301	5.9481 ± 0.2481	1.3899 ± 0.0197
kNN	0.9845 ± 0.0187	0.7939 ± 0.0096	0.8121 ± 0.0125	0.7951 ± 0.0121	0.7699 ± 0.0157	$\mathbf{0.8413 \pm 0.0122}$	0.7794 ± 0.0094
LR	0.8925 ± 0.0170	0.8659 ± 0.0090	0.7776 ± 0.0116	0.7208 ± 0.0099	0.7080 ± 0.0128	1.0219 ± 0.0161	0.7366 ± 0.0095
Ours (no dc)	0.9451 ± 0.0164	$\mathbf{0.7878 \pm 0.0095}$	0.7450 ± 0.0109	$\mathbf{0.6767 \pm 0.0094}$	0.7147 ± 0.0119	0.9922 ± 0.0227	$\mathbf{0.7336 \pm 0.0094}$
Ours	$\mathbf{0.8880 \pm 0.0160}$	0.7991 ± 0.0107	$\mathbf{0.7433 \pm 0.0105}$	0.6808 ± 0.0101	$\mathbf{0.6741 \pm 0.0107}$	0.9253 ± 0.0166	0.7564 ± 0.0136
Input-agnostic	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 5. Grasp collision rate (↓) / mesh-IoU (↑) for each reconstruction model across 10 YCB objects. Grasps are generated on the reconstructed mesh using a two-fingered gripper [23] and evaluated against the ground-truth mesh. SCOUT is evaluated in two settings: routing over novel views of known objects, and routing over both novel views and novel objects.

Object	Hunyuan3D	InstantMesh	TRELLIS	TripoSR	Ours (novel obj.)	Ours (novel view)
Tomato Soup Can	0.7174 / 0.0685	0.0227 / 0.9164	0.3864 / 0.2071	0.6585 / 0.1807	InstantMesh	InstantMesh
Cup (vC)	0.0 / 0.4019	0.2273 / 0.1381	0.0 / 0.4372	0.1500 / 0.1487	Hunyuan3D	Hunyuan3D
Marble	0.0 / 0.9580	0.0 / 0.7423	0.0 / 0.1464	0.0 / 0.9537	TripoSR	TripoSR
Chef Can	0.0 / 0.8543	0.0 / 0.8763	0.0250 / 0.9406	0.1190 / 0.7730	InstantMesh	TRELLIS
Potted Meat	0.0 / 0.6589	0.6098 / 0.4604	0.0682 / 0.7573	0.3571 / 0.4632	InstantMesh	TRELLIS
Cup (vB)	0.0 / 0.2960	0.0238 / 0.1423	0.1087 / 0.0338	0.3659 / 0.1080	Hunyuan3D	Hunyuan3D
Tennis Ball	0.0 / 0.9717	0.0 / 0.9663	0.0 / 0.1525	0.0 / 0.9645	InstantMesh	InstantMesh
Brick	0.0652 / 0.2421	0.0652 / 0.2547	0.6136 / 0.0949	0.0 / 0.8334	TripoSR	TripoSR
Power Drill	0.0714 / 0.6230	0.0500 / 0.5662	0.2889 / 0.1399	0.3636 / 0.4793	InstantMesh	InstantMesh
Spatula	0.0851 / 0.2218	0.0465 / 0.0076	0.0851 / 0.1014	0.0 / 0.0392	InstantMesh	Hunyuan3D
Mean	0.0939 / 0.5296	0.1045 / 0.5071	0.1576 / 0.3011	0.2014 / 0.4944	0.0729 / 0.6278	0.0251 / 0.6854

Table 6. Dexterous manipulation success rate (↑) in simulation for each reconstruction model across 5 objects (HY3D: Hunyuan3D, IM: InstantMesh, TRL: TRELLIS, TSR: TripoSR). Ours reports the model selected by SCOUT.

Method	Can	Spatula	Drill	Brick	Cup	Mean
HY3D	13.6	15.2	38.1	61.8	49.4	35.6
IM	29.9	0	41.7	52.9	44.5	33.8
TRL	61.8	0	4.5	50.4	45.8	32.5
TSR	21.9	0	17.6	39.7	45.0	24.8
Ours	TRL	HY3D	IM	TSR	HY3D	41.6

Table 7. Real-world pick-and-place success rate (↑) on a Franka Panda robot across 5 objects. For each reconstructed object, 5 antipodal grasps are sampled and executed.

Method	Drill	Ball	Can	Cup	Mug	Mean
HY3D	3/5	4/5	5/5	5/5	5/5	4.4/5
IM	1/5	5/5	1/5	2/5	4/5	2.6/5
TRL	0/5	5/5	3/5	0/5	5/5	2.6/5
TSR	1/5	0/5	2/5	0/5	0/5	0.6/5
Ours	HY3D	TRL	HY3D	HY3D	TRL	4.6/5

6. Acknowledgments

We gratefully acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR MURI grants N00014-22-1-2740 and N00014-24-1-2603; from the MIT Siegel Family Quest for Intelligence; and from the Robotics and AI Institute.

References

[1] Aditya Agarwal, Gaurav Singh, Bipasha Sen, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Scenecomplete: Open-world 3d scene completion in cluttered real world environments for robot manipulation. *IEEE Robotics and Automation Letters*, 11(1):482–489, 2026. 1, 7

[2] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. 2, 5, 14

[3] Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024. 1, 2, 5

[4] Alexandre Devert. miniball: Efficient computation of the smallest bounding ball of a point set. https://github.com/marmakoide/miniball, 2023. Python implementation of Welzl's algorithm. 14

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6, 13

[6] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. Ieee, 2022. 2, 4, 5, 14

[7] Shivam Duggal, Yushi Hu, Oscar Michel, Aniruddha Kembhavi, William T Freeman, Noah A Smith, Ranjay Krishna, Antonio Torralba, Ali Farhadi, and Wei-Chiu Ma. Eval3d: Interpretable and fine-grained evaluation for 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13326–13336, 2025. 2, 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 13

[9] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023. 1, 2

[10] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024. 1, 3, 5, 6

[11] Shun Iwase, Katherine Liu, Vitor Guizilini, Adrien Gaidon, Kris Kitani, Rareş Ambruş, and Sergey Zakharov. Zero-shot multi-object scene completion. In *European Conference on Computer Vision*, pages 96–113. Springer, 2024. 1

[12] Shun Iwase, Muhammad Zubair Irshad, Katherine Liu, Vitor Guizilini, Robert Lee, Takuya Ikeda, Ayako Amma, Koichi Nishiwaki, Kris Kitani, Rares Ambrus, et al. Zerograsp: Zero-shot shape reconstruction enabled robotic grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17405–17415, 2025. 1

[13] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, et al. Universal model routing for efficient llm inference. *arXiv preprint arXiv:2502.08773*, 2025. 3

[14] Abhishek Kashyap, Yuxuan Yang, Henrik Andreasson, and Todor Stoyanov. Single-view shape completion for robotic grasping in clutter. *arXiv preprint arXiv:2512.16449*, 2025. 1

[15] Yang Li. Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing. *arXiv preprint arXiv:2502.02743*, 2025. 3

[16] Yang Li. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025. 1, 3, 5, 14

[17] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36:22226–22246, 2023. 1, 2

[18] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10072–10083, 2024. 2

[19] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 1, 2, 7

[20] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 6, 13

[21] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974, 2024. 1, 2, 4, 5

[22] Jens Lundell, Francesco Verdoja, and Ville Kyrki. Beyond top-grasps through scene completion. In *2020 IEEE Inter-*

national Conference on Robotics and Automation (ICRA), pages 545–551. IEEE, 2020. 1

[23] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019. 7, 8

[24] Seyed S Mohammadi, Nuno F Duarte, Dimitris Dimou, Yiming Wang, Matteo Taiana, Pietro Morerio, Atabak Dehban, Plinio Moreno, Alexandre Bernardino, Alessio Del Bue, et al. 3dsgrasp: 3d shape-completion for robotic grasp. *arXiv preprint arXiv:2301.00866*, 2023. 1

[25] Frederik Nolte, Andreas Geiger, Bernhard Schölkopf, and Ingmar Posner. Is single-view mesh reconstruction ready for robotics? *arXiv preprint arXiv:2505.17966*, 2025. 1

[26] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024. 1, 5

[27] Guanzhong Pan and Haibo Wang. A cost-benefit analysis of on-premise large language model deployment: Breaking even with commercial llm services. *arXiv preprint arXiv:2509.18101*, 2025. 3

[28] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 1, 2

[29] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023. 1, 2

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 6, 13

[31] Siyu Ren, Junhui Hou, Xiaodong Chen, Hongkai Xiong, and Wenping Wang. Ddm: A metric for comparing 3d shapes using directional distance fields. *arXiv preprint arXiv:2401.09736*, 2024. 5, 14

[32] Bipasha Sen, Aditya Agarwal, Gaurav Singh, Srinath Sridhar, Madhava Krishna, et al. Scarp: 3d shape completion in arbitrary poses for improved grasping. *arXiv preprint arXiv:2301.07213*, 2023. 1

[33] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Hyeonsu Kim, Jaehoon Ko, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. *arXiv preprint arXiv:2303.07937*, 2023. 1, 2

[34] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 1, 2

[35] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014. 2, 5, 14

[36] Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025. 1, 2, 5, 14

[37] Tencent Hunyuan Team. Hunyuan3D-2 source code repository. https://github.com/Tencent-Hunyuan/Hunyuan3D-2, 2025. 14

[38] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image, 2024. 1, 2, 3, 5, 14

[39] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. TripoSR: Fast 3d object reconstruction from a single image. https://github.com/VAST-AI-Research/TripoSR, 2024. 14

[40] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12619–12629, 2023. 1, 2

[41] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 14

[42] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17868–17879, 2024. 5, 14

[43] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702*, 2021. 2, 5

[44] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. TRELLIS: Structured 3d latents for scalable and versatile 3d generation. https://github.com/microsoft/TRELLIS, 2024. 14

[45] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. 1, 2, 5, 14

[46] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. https://github.com/TencentARC/InstantMesh, 2024. 14

[47] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models, 2024. 1, 2, 3, 5, 14

[48] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing

the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10371–10381, 2024. 7

Which Reconstruction Model Should a Robot Use? Routing Image-to-3D Models for Cost-Aware Robotic Manipulation

Supplementary Material

7. Derivation of optimal weights

We derive global inverse-variance weights for combining k_1 estimators of the partition function. Each method $j \in \{1, \dots, k_1\}$ produces, for data point i, the estimate

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}, \tag{8}$$

and we seek weights w_j, independent of i, for the combined estimate

$$\tilde{z}^{(i)} = \sum_{j=1}^{k_1} w_j \, \tilde{z}_j^{(i)}. \tag{9}$$

Assumptions.

A1. (Error model). For each method j and data point i, $\hat{p}_j^{(i)} = p_j^{*(i)} + \epsilon_j^{(i)}$, where $p_j^{*(i)}$ is the true (fixed) probability and $\epsilon_j^{(i)}$ is random noise with $\mathbb{E}_\epsilon[\epsilon_j^{(i)}] = 0$ and $\mathrm{Var}_\epsilon[\epsilon_j^{(i)}] = \sigma_j^2$. The noise variance σ_j^2 is a property of method j alone and does not depend on i.

A2. (Small noise). $|\epsilon_j^{(i)}| \ll p_j^{*(i)}$ for all j, i.

A3. (Uncorrelated methods). $\mathrm{Cov}_\epsilon[\epsilon_j^{(i)}, \epsilon_k^{(i)}] = 0$ for $j \neq k$.

A4. (Consistency). All methods estimate the same true quantity: $z^{(i)} \equiv e^{s_j^{(i)}/T} / p_j^{*(i)}$ is independent of j.

A5. (Noise–signal independence). $\epsilon_j^{(i)}$ is independent of $p_j^{*(i)}$ across data points; *i.e.*, the estimation error does not depend on the true probability.

Here $\mathbb{E}_\epsilon$ and Var_ϵ denote expectation and variance over estimation noise at fixed i; all other quantities are deterministic for a given i.

Global weight optimization. Fixing a data point i and expanding Eq. (8) to first order in $\epsilon_j^{(i)}/p_j^{*(i)}$:

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{p_j^{*(i)} + \epsilon_j^{(i)}} \approx z^{(i)} - \frac{z^{(i)}}{p_j^{*(i)}} \epsilon_j^{(i)}. \tag{10}$$

Hence

$$\mathrm{Var}_\epsilon[\tilde{z}_j^{(i)}] = \left(\frac{z^{(i)}}{p_j^{*(i)}}\right)^2 \sigma_j^2. \tag{11}$$

Substituting $p_j^{*(i)} = e^{s_j^{(i)}/T}/z^{(i)}$ from **A4**:

$$\tau_j^{2,(i)} \equiv \mathrm{Var}_\epsilon[\tilde{z}_j^{(i)}] = \frac{\left(z^{(i)}\right)^4}{e^{2s_j^{(i)}/T}} \cdot \sigma_j^2. \tag{12}$$

Inserting Eq. (10) into Eq. (9):

$$\tilde{z}^{(i)} - z^{(i)} \approx -\sum_j w_j \frac{z^{(i)}}{p_j^{*(i)}} \epsilon_j^{(i)}. \tag{13}$$

Taking the variance over the joint noise $\{\epsilon_j^{(i)}\}_j$ and using **A3**:

$$\mathrm{Var}_\epsilon[\tilde{z}^{(i)}] = \sum_j w_j^2 \, \tau_j^{2,(i)}. \tag{14}$$

The quantity Eq. (14) depends on i through $\tau_j^{2,(i)}$. Since we seek weights independent of i, we minimize the variance *averaged* over data points:

$$\begin{aligned} J(\mathbf{w}) &= \mathbb{E}_i\big[\mathrm{Var}_\epsilon[\tilde{z}^{(i)}]\big] \\ &= \mathbb{E}_i\Big[\sum_j w_j^2 \, \tau_j^{2,(i)}\Big] \\ &= \sum_j w_j^2 \, \mathbb{E}_i[\tau_j^{2,(i)}] \\ &\equiv \sum_j w_j^2 \, \bar{\tau}_j^{\,2} \end{aligned} \tag{15}$$

Substituting Eq. (12):

$$\bar{\tau}_j^{\,2} = \sigma_j^2 \, \mathbb{E}_i\left[\frac{\left(z^{(i)}\right)^4}{e^{2s_j^{(i)}/T}}\right], \tag{16}$$

We minimize Eq. (15) subject to $\sum_j w_j = 1$. The Lagrangian is

$$\mathcal{L} = \sum_j w_j^2 \, \bar{\tau}_j^{\,2} - \lambda\left(\sum_j w_j - 1\right). \tag{17}$$

Setting $\partial\mathcal{L}/\partial w_j = 2\, w_j \, \bar{\tau}_j^{\,2} - \lambda = 0$ gives $w_j = \lambda/(2\bar{\tau}_j^{\,2})$. Substituting into the constraint:

$$\sum_j \frac{\lambda}{2\bar{\tau}_j^{\,2}} = 1 \quad \Longrightarrow \quad \lambda = \frac{2}{\sum_k 1/\bar{\tau}_k^{\,2}}. \tag{18}$$

The Hessian of $\mathcal{L}$ restricted to the constraint surface has diagonal entries $2\bar{\tau}_j^{\,2} > 0$, confirming this is a minimum. The optimal global weights are

$$\boxed{w_j = \frac{1/\bar{\tau}_j^{\,2}}{\sum_k 1/\bar{\tau}_k^{\,2}} \propto \frac{1}{\bar{\tau}_j^{\,2}} = \frac{1}{\sigma_j^2 \, \mathbb{E}_i\left[\frac{(z^{(i)})^4}{e^{2s_j^{(i)}/T}}\right]}}. \tag{19}$$

Estimation of σ_j^2. It remains to estimate the noise variance σ_j^2 from data. Define the following quantities computed across data points:

$$S_j^2 \equiv \mathrm{Var}_i\big[\hat{p}_j^{(i)}\big], \qquad R_j^2 \equiv \mathrm{Corr}_i\big[p_j^{*(i)},\, \hat{p}_j^{(i)}\big]^2. \quad (20)$$

Using the error model $\hat{p}_j^{(i)} = p_j^{*(i)} + \epsilon_j^{(i)}$, the independence assumption **A5** gives $\mathrm{Cov}_i[p_j^*,\, \epsilon_j] = 0$, and since σ_j^2 is constant across i (**A1**), we have:

$$S_j^2 = \mathrm{Var}_i\big[p_j^{*(i)} + \epsilon_j^{(i)}\big] = \mathrm{Var}_i\big[p_j^{*(i)}\big] + \sigma_j^2. \quad (21)$$

Using $\mathrm{Cov}_i[p_j^*,\, \hat{p}_j] = \mathrm{Cov}_i[p_j^*,\, p_j^* + \epsilon_j] = \mathrm{Var}_i[p_j^*]$, we find:

$$R_j^2 = \frac{\mathrm{Cov}_i[p_j^*,\, \hat{p}_j]^2}{\mathrm{Var}_i[p_j^*] \cdot S_j^2} = \frac{\mathrm{Var}_i[p_j^*]^2}{\mathrm{Var}_i[p_j^*] \cdot S_j^2} = \frac{\mathrm{Var}_i[p_j^*]}{S_j^2}. \quad (22)$$

Substituting $\mathrm{Var}_i[p_j^{*(i)}] = R_j^2\, S_j^2$ into Eq. (21):

$$\boxed{\sigma_j^2 \;=\; S_j^2\,(1 - R_j^2).} \quad (23)$$

8. Effect of smoothing on score recovery

In Section 3.2, we train $\hat{\mathbf{p}}$ on the softmax of the smoothed scores $\tilde{\mathbf{s}}_{\mathrm{dep}}^{(i)}$, but the recovered scores $\hat{s}(x^{(i)}, m_j) = T \cdot \ln\big(\hat{p}_j^{(i)} \cdot \hat{z}^{(i)}\big)$ should approximate the true unsmoothed scores $s_{\mathrm{dep},j}^{(i)}$ for comparison with $\mathbf{s}_{\mathrm{inv}}$. Here we show that the proxy construction targets these true scores by design. Throughout this section we write $s_j^{(i)} \equiv s_{\mathrm{dep},j}^{(i)}$ for brevity.

Role of smoothing. The purpose of tag-based smoothing is to reduce noise in the training targets, producing more accurate predictions $\hat{\mathbf{p}}$ of relative model performance—the smoothed scores $\tilde{\mathbf{s}}_{\mathrm{dep}}^{(i)}$ are not themselves the quantities we wish to recover. The ablation in Tab. 8 confirms that smoothing improves routing performance, validating this role. Smoothing is not applied when constructing the proxy targets for $\hat{z}$, because $\hat{z}$ must capture the absolute difficulty of each individual input image, and smoothing suppresses precisely this per-image variation by blending scores toward category averages. This is consistent with the baseline ablation (Tab. 9): applying smoothing to LR and kNN, which predict absolute scores rather than relative probabilities, degrades their performance.

Table 8. Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset.

Method	$\{\mathbf{c} \in \mathcal{C}_0\}$	$\{\mathbf{c} \in \mathcal{C}\}$	$\{\mathbf{c} \in \mathcal{C} \cap \mathbf{c}_{\mathrm{inv}} = \infty\}$
Ours w/ smoothing	0.6386 ± 0.0033	0.4319 ± 0.0021	0.4831 ± 0.0019
Ours w/o smoothing	0.6588 ± 0.0035	0.4371 ± 0.0021	0.4878 ± 0.0020

Table 9. Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset.

Method	$\{\mathbf{c} \in \mathcal{C}_0\}$	$\{\mathbf{c} \in \mathcal{C}\}$	$\{\mathbf{c} \in \mathcal{C} \cap \mathbf{c}_{\mathrm{inv}} = \infty\}$
LR w/ smoothing	0.6515 ± 0.0034	0.4723 ± 0.0022	0.5064 ± 0.0019
LR w/o smoothing	0.6497 ± 0.0033	0.4519 ± 0.0021	0.5047 ± 0.0019
kNN w/ smoothing	0.6515 ± 0.0034	0.4723 ± 0.0022	0.5064 ± 0.0019
kNN w/o smoothing	0.6489 ± 0.0033	0.4630 ± 0.0022	0.4992 ± 0.0019

Proxy construction targets true scores. Given $\hat{\mathbf{p}}$, we wish to find $\hat{z}^{(i)}$ such that $T \cdot \ln\big(\hat{p}_j^{(i)} \cdot \hat{z}^{(i)}\big) = s_j^{(i)}$. Solving for each model j individually gives the per-model proxy:

$$\tilde{z}_j^{(i)} \;=\; \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}, \quad (24)$$

which uses the true unsmoothed score $s_j^{(i)}$ in the numerator. By construction:

$$T \cdot \ln\big(\hat{p}_j^{(i)} \cdot \tilde{z}_j^{(i)}\big) \;=\; T \cdot \ln\left(\hat{p}_j^{(i)} \cdot \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}\right) \;=\; s_j^{(i)}. \quad (25)$$

Each proxy exactly recovers the true score for model j, regardless of what $\hat{p}_j^{(i)}$ was trained on, because the proxy is defined directly from the true scores.

9. Implementation details

SCOUT implementation details. The probability network $\hat{\mathbf{p}}$ is a feedforward neural network with two hidden layers of 128 units each, ReLU activations, and dropout ($p = 0.2$), trained with KL divergence loss using Adam (learning rate 10^{-4}, weight decay 10^{-4}) with batch size 128 for 10 epochs. Tag smoothing uses $\beta = 0.7$ and softmax temperature $T = 1.0$ for the GSO dataset. For BigBIRD + YCB, we decrease β to 0.5 because the scores are noisier, requiring stronger smoothing toward category averages. The softmax temperature T is set per metric to account for differences in score scale (Tab. 10).

Table 10. Softmax temperature T per metric for BigBIRD + YCB.

Metric	**DCD**	**Chamfer L2**	**Chamfer L1**	**IoU**	**MMD-EMD**	**Eval3D-geo**	**Eval3D-struct**
T	0.5	0.1	0.2	2.0	0.15	0.1	0.2

The partition function $\hat{z}$ is predicted by Ridge regression ($\alpha = 10^3$), with the weighting terms R_j^2, S_j^2, and the expectation in w_j estimated via 5-fold cross-validation over the training set. Image features are concatenated from CLIP ViT-B/32 (512-d) [30], ResNet-50 (2048-d) [8], ViT-B/16 (768-d) [5], and ConvNeXt-B (1024-d) [20], yielding a 4352-dimensional input. Hyperparameters were selected on seeds 0-49. All final experiments were conducted on a

single NVIDIA RTX 4090 GPU and averaged over seeds 50-199.

Data collection. Scores were collected as follows. We began with ground-truth meshes from the GSO [6] and YCB [2] datasets, and collected images of each mesh at multiple viewpoints. For the GSO dataset, images were rendered at elevation angles of $35°, 45°, 60°, 75°$, and $85°$. All elevations were captured at azimuth angles of $30°, 120°$, and $210°$. This combination ensures all captured views are nondegenerate. For the YCB dataset, images were captured at elevation angles of $5°, 22.5°, 45°, 67.5°$, and $90°$. The top view ($90°$) was captured at a single azimuth angle of $45°$, while all other elevations were captured at azimuth angles of $0°, 45°$, and $90°$. Unlike GSO, this combination of elevation and azimuth angles produces many degenerate viewpoints. Each YCB viewpoint was captured under three conditions: flash-style lighting, surround-style lighting, and a real image from the BigBIRD dataset [35]. Examples of degenerate versus nondegenerate views are shown in Fig. 4 and examples of different lighting styles are shown in Fig. 5. Surround-style lighting is typically more challenging for Image-to-3D models than flash-style lighting, as the lack of shadows removes geometric cues that aid reconstruction.

(a) Degenerate view showing only the front face. (b) Nondegenerate view revealing full geometry. (c) Degenerate top-down view.

Figure 4. Degenerate and nondegenerate views of a cracker box.

(a) Rendered image with flash-style lighting. (b) Rendered image with surround-style lighting. (c) Real image from BigBIRD [35].

Figure 5. Different lighting conditions for a fixed viewpoint.

Once images were collected, we generated a reconstruction with each of the four Image-to-3D models: Hunyuan3D [36] (version 2.0 throughout), InstantMesh [47], TRELLIS [45], and TripoSR [38], using default parameters for all models. For Hunyuan3D, we applied a convex decomposition using CoACD [41] (concavity threshold 0.025, 50 MCTS search iterations, default remaining parameters) between the mesh generation and texture generation stages to prevent the texturing process from taking excessive time.

The ground-truth and reconstructed meshes were then normalized to a common scale by fitting each mesh to a unit sphere centered at the origin using *miniball* [4]. We then registered each reconstructed mesh to its ground-truth counterpart via FoundationPose [42] for global registration (default parameters), followed by DDM [31] for local refinement (learning rate 2e-2, 100 iterations).

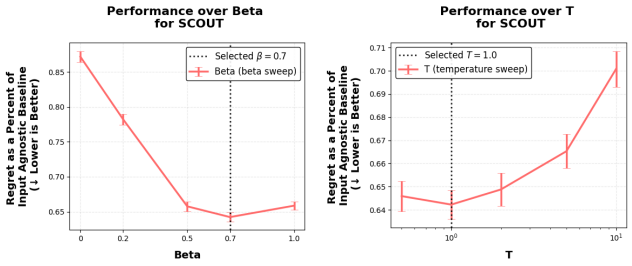
Cost vectors. The AIQ and deferral curve metrics require latency, memory, and latency$\odot$memory cost vectors. Latency costs are 30s (Hunyuan3D), 10s (InstantMesh), 10s (TRELLIS), and 0.5s (TripoSR). Memory costs are 6GB (Hunyuan3D), 24GB (InstantMesh), 16GB (TRELLIS), and 6GB (TripoSR). All values reflect mesh generation only (excluding texturing) and are based on official documentation [37, 39, 44, 46] when available, otherwise approximated by experiments.

10. Ablations

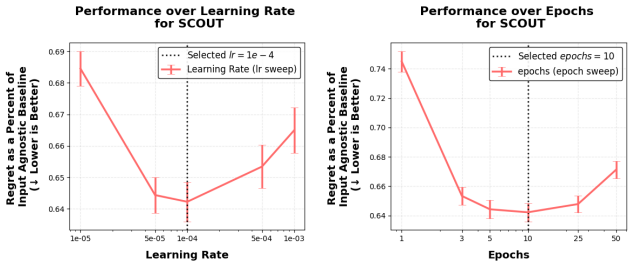
SCOUT hyperparameters. We perform one-at-a-time sweeps over each hyperparameter, holding the others fixed, to verify that the loss landscape is smooth with a local optimum near our selected values. These experiments were conducted on seeds 0-49. The network architecture (two hidden layers of 128 units each) was fixed to ensure fair comparison with the baselines. For the GSO dataset, the selected hyperparameters are $\beta = 0.7$, $T = 1.0$, learning rate 10^{-4}, and 10 training epochs; smooth behavior across all four hyperparameters is shown in Fig. 6. These hyperparameters govern the $\hat{\mathbf{p}}$ model. The $\hat{z}$ model uses Ridge regression with regularization parameter $\alpha = 10^3$, which similarly exhibits smooth behavior (Fig. 7).

We repeat this analysis for SCOUT without decoupling, using 25 epochs instead of 10, and find a similarly smooth landscape with the same remaining hyperparameters (Fig. 8). The increase in epochs is expected: without decoupling, the $\hat{\mathbf{p}}$ network must capture both relative model performance and overall image difficulty, requiring more training to converge.

Baseline hyperparameters. Since both our work and Li [16] find that kNN and LR models are among the strongest baselines, we analyze their hyperparameter sweeps as well. Both exhibit similarly smooth landscapes (kNN: Fig. 9; LR: Fig. 10).

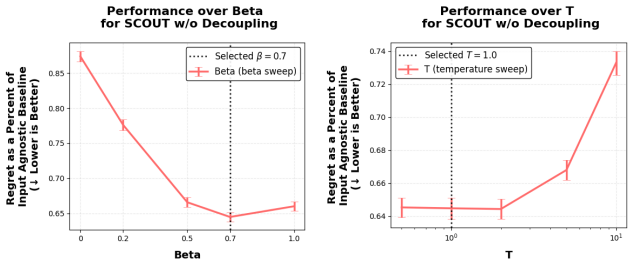


(a) Beta sweep. (b) Temperature sweep.

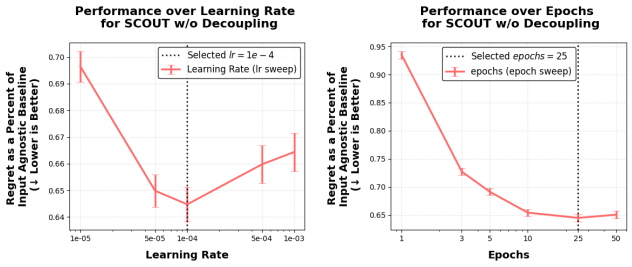


(c) Learning rate sweep. (d) Epoch sweep.

Figure 6. Sensitivity of hyperparameters for SCOUT evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.



(a) Beta sweep. (b) Temperature sweep.



(c) Learning rate sweep. (d) Epoch sweep.

Figure 8. Sensitivity of hyperparameters for SCOUT without decoupling evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

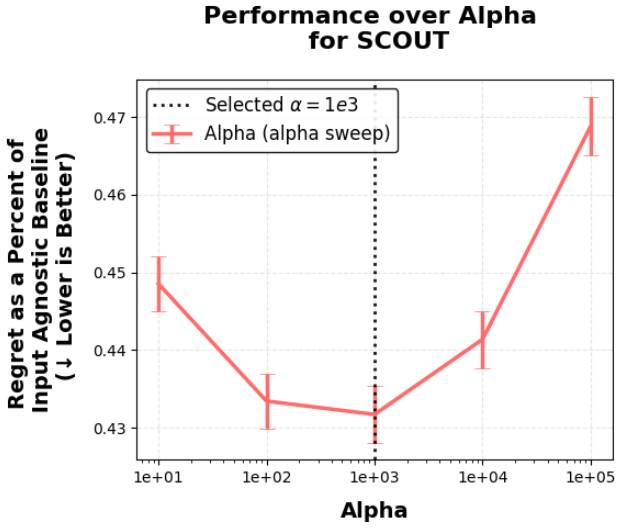


Figure 7. Sensitivity of alpha for SCOUT evaluated on novel objects in the GSO dataset over cost coefficient vectors sampled from $\mathcal{C}$.

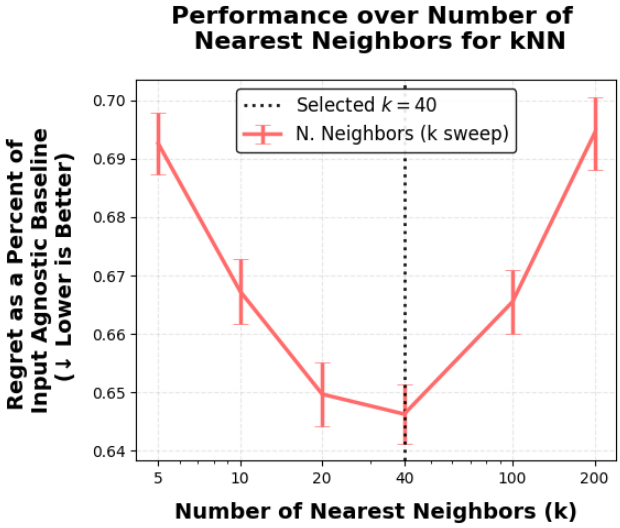


Figure 9. Sensitivity of the number of nearest neighbors for the kNN baseline evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

11. Additional results

11.1. Weighted proxy results.

Our weighted proxy improves over all three alternatives—the ground-truth $z^{(i)}$, equal weighting, and one-hot weighting ($\tilde{z}^{(i)} = \tilde{z}^{(i)}_{j^*}$ where $j^* = \arg\max_j w_j$)—on the majority of metrics, as confirmed by one-sided t-tests (Tab. 11). Most notably, it outperforms the ground-truth $z^{(i)}$ on 6 of 7 metrics with no deteriorations. The ground-truth $z^{(i)}$ is computed as a sum of exponentials over noisy reconstruction scores, so models with highly variable scores disproportionately inflate its variance. The derived weights mitigate this by downweighting model j when (1) its predicted probabilities $\hat{p}_j$ are noisier (large $S_j^2(1 - R_j^2)$), or (2) its

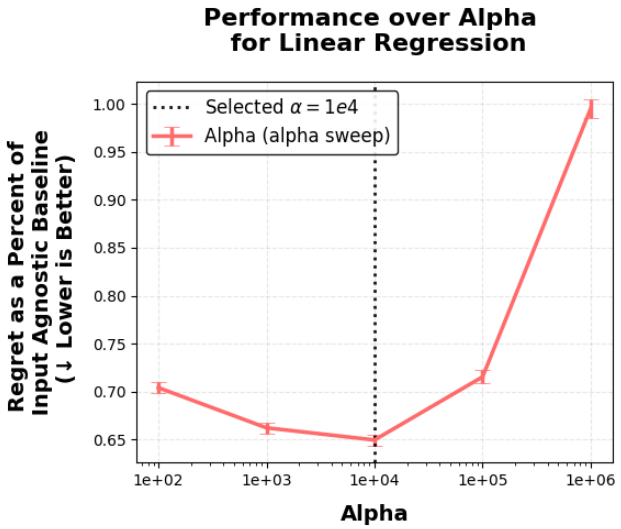


Figure 10. Sensitivity of alpha for the linear regression baseline evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

true probability is frequently low, making the proxy $\tilde{z}_j^{(i)}$ a higher-variance estimate of $z^{(i)}$ (see Eq. (11)).

Table 11. One-sided t-test p-values comparing our weighted partition function proxy against three alternatives on BigBIRD + YCB. Each cell reports the p-value for the hypothesis that our proxy achieves lower regret, averaged over cost coefficient vectors sampled from $\mathcal{C}$ (Sec. 4.1). An improvement indicates a metric where our proxy achieves statistically significant ($\alpha = 0.05$) lower regret than the alternative; a deterioration ($\alpha = 0.05$ for the reverse one-sided test) indicates metrics where the alternative achieves statistically significant lower regret than ours.

Metric	**True z**	**Equiweighted**	**One-hot**
DCD	3.24e-4	5.44e-1	2.38e-24
Chamfer L2	4.01e-35	5.74e-35	8.11e-1
Chamfer L1	2.03e-4	3.92e-2	7.71e-16
IoU	4.35e-1	2.31e-1	6.89e-5
MMD-EMD	5.48e-13	3.32e-6	4.10e-4
Eval3D-geo	8.24e-34	2.70e-30	9.09e-4
Eval3D-struct	7.78e-18	2.31e-18	9.95e-1
# improvements	6/7	5/7	5/7
# deteriorations	0/7	0/7	1/7

11.2. Grasp collisions

Figure 11 illustrates two routing decisions chosen by SCOUT. For a top-down view of a can, only InstantMesh produces a usable reconstruction, and SCOUT selects it. For a frontal view of a meat container, SCOUT selects TRELLIS, which reconstructs the object accurately at a lower latency than Hunyuan3D. These examples show that per-input routing can improve reconstruction quality and reduce latency.

Figures 12 and 13 show the eight remaining objects from the grasp collision experiment in Tab. 5. Across these objects, SCOUT routes to a variety of reconstruction models depending on the input viewpoint, confirming that no single model dominates and that per-input selection is beneficial.

11.3. Policy learned

To understand what SCOUT learns, we examine which model is selected as a function of object category (Fig. 14) and input viewpoint (Fig. 15). Different reconstruction models exhibit distinct biases across categories and viewpoints, and SCOUT's learned policy reflects these trends even when evaluating on novel objects, confirming that the router captures category-dependent biases and viewpoint-dependent biases from the training data.

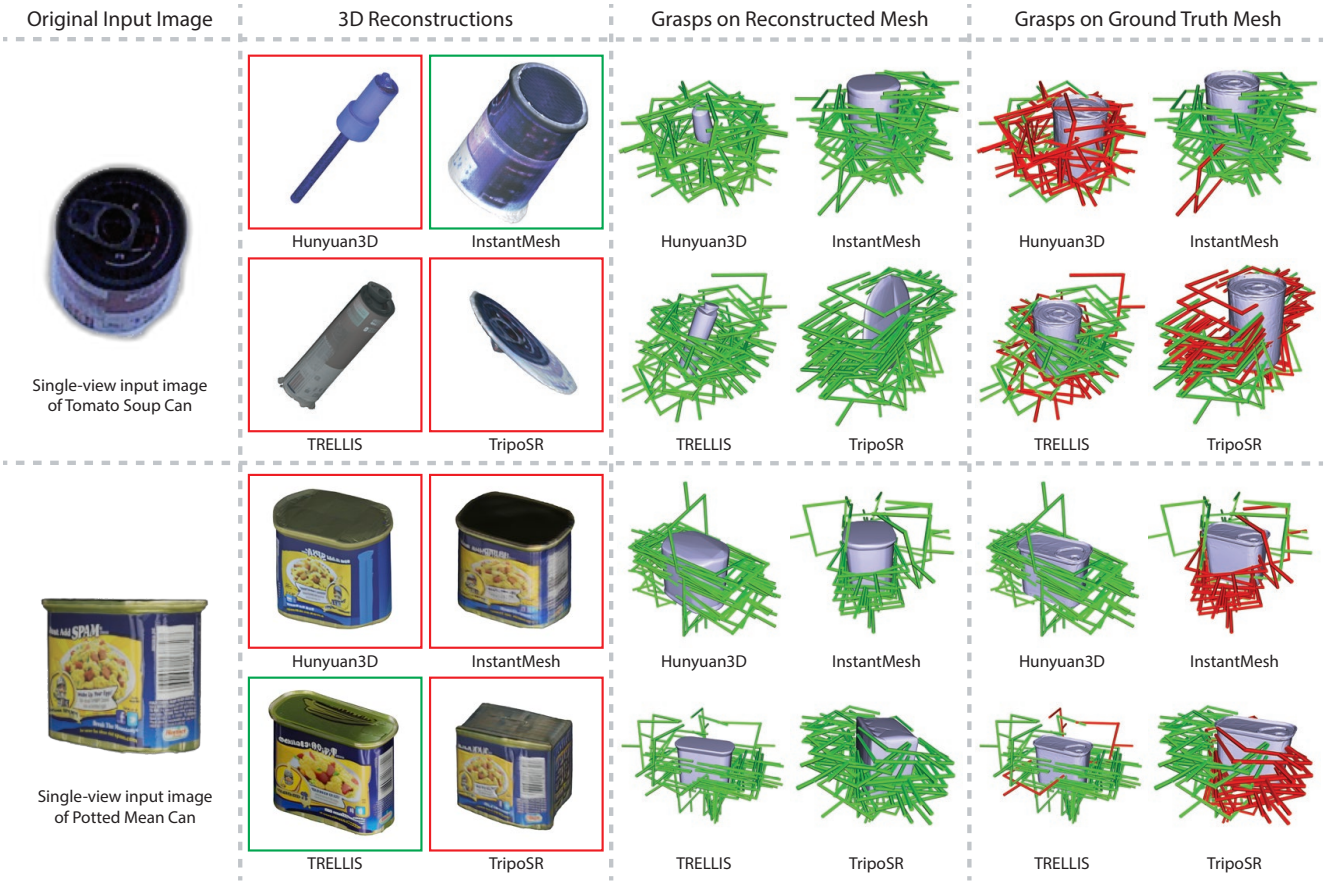


Figure 11. Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red.

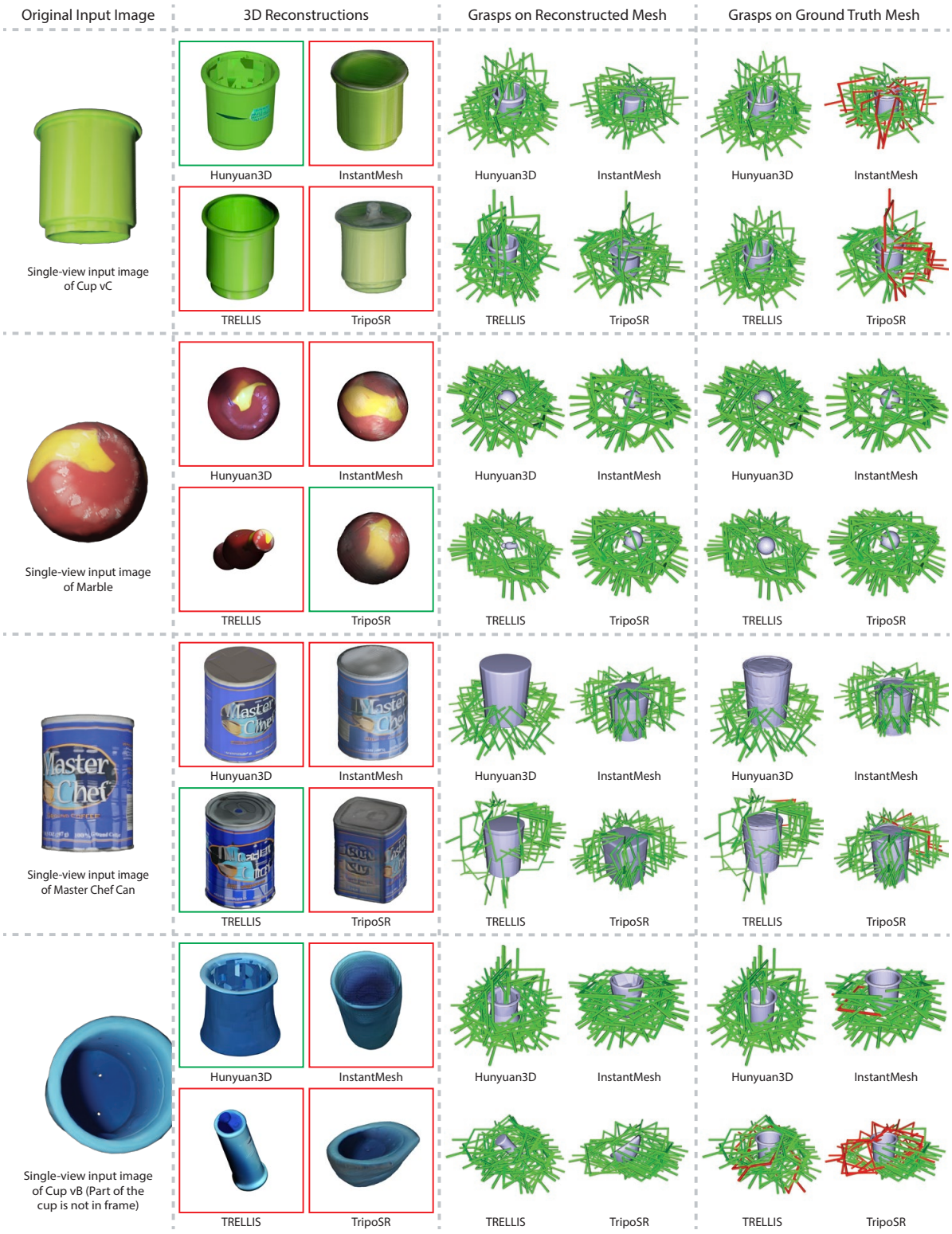


Figure 12. Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red.

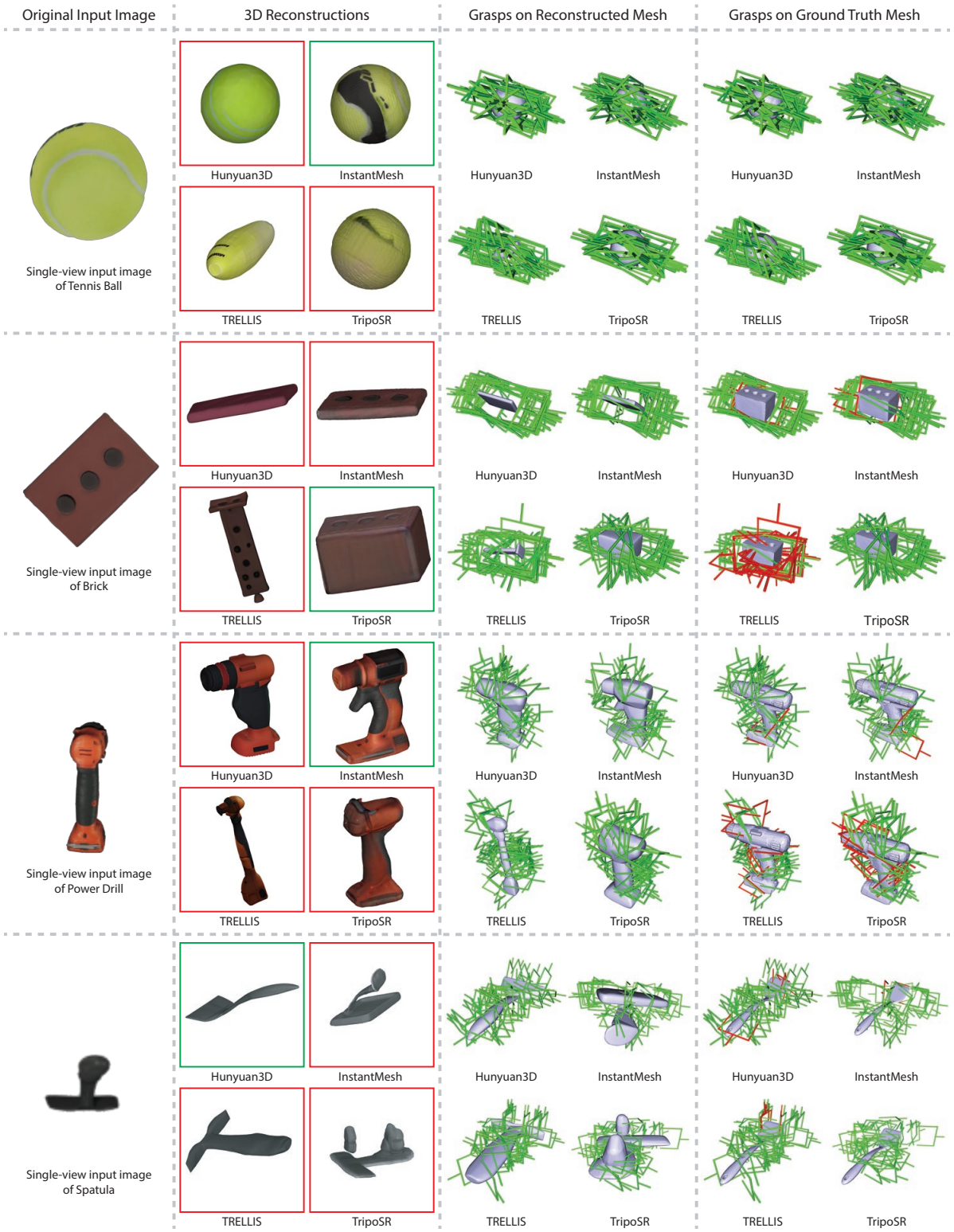
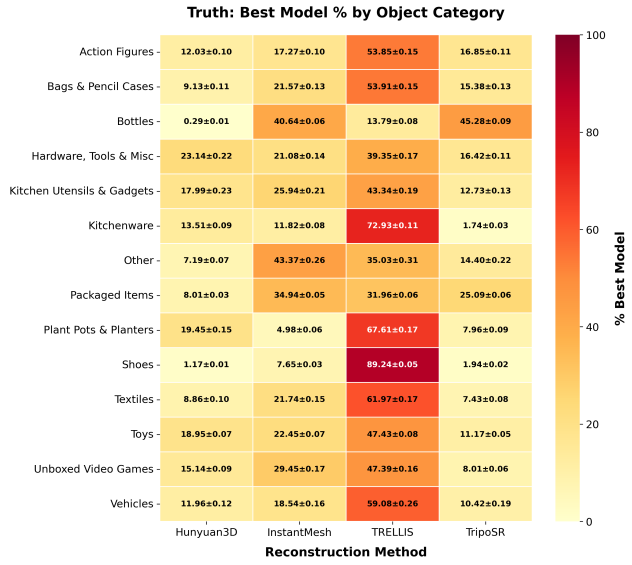
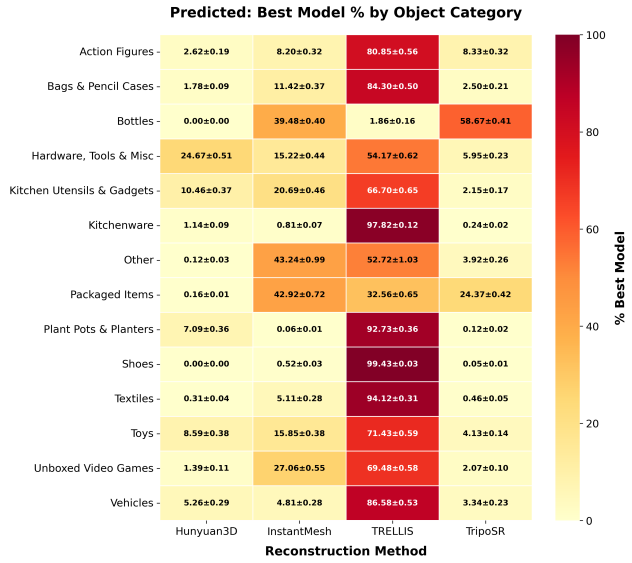


Figure 13. Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red.

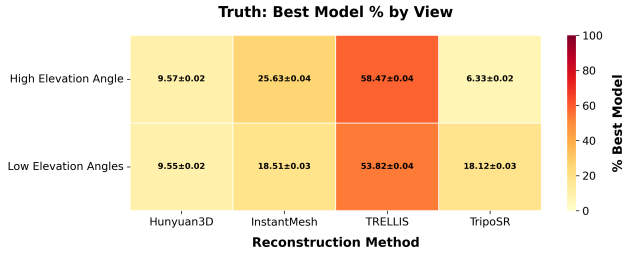


(a) Oracle policy: how often each model is truly optimal when evaluated on unseen objects, grouped by object category.

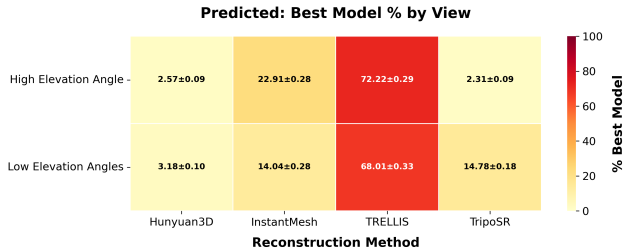


(b) SCOUT's learned policy: how often each model is selected when evaluated on unseen objects, grouped by object category.

Figure 14. Oracle and learned routing policies across object categories.



(a) Oracle policy: how often each model is truly optimal when evaluated on unseen objects, grouped by elevation.



(b) SCOUT's learned policy: how often each model is selected when evaluated on unseen objects, grouped by elevation.

Figure 15. Oracle and learned routing policies across viewpoint elevations. High: 75° and 85°; low: 35°, 45°, and 60°.