Input-Aware Routing of Image-to-3D Models for Robotic Manipulation

Supplementary Material

I. 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

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}, \tag{1}$$

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

$$\tilde{z}^{(i)} = \sum_{j=1}^{k_1} w_j \, \tilde{z}_j^{(i)}. \tag{2}$$

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 $\mathbb{E}_\epsilon\big[\epsilon_j^{(i)}\big] = 0$ and $\mathrm{Var}_\epsilon\big[\epsilon_j^{(i)}\big] = \sigma_j^2$. 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).** $\mathrm{Cov}_\epsilon\big[\epsilon_j^{(i)}, \epsilon_k^{(i)}\big] = 0$ 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. (1) to first order in $\epsilon_j^{(i)}/p_j^{*(i)}$:

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{p_j^{*(i)} + \epsilon_j^{(i)}} \;\approx\; z^{(i)} \;-\; \frac{z^{(i)}}{p_j^{*(i)}}\,\epsilon_j^{(i)}. \tag{3}$$

Hence

$$\mathrm{Var}_\epsilon\big[\tilde{z}_j^{(i)}\big] = \left(\frac{z^{(i)}}{p_j^{*(i)}}\right)^2 \sigma_j^2. \tag{4}$$

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

$$\tau_j^{2,(i)} \;\equiv\; \mathrm{Var}_\epsilon\big[\tilde{z}_j^{(i)}\big] \;=\; \frac{\big(z^{(i)}\big)^4}{e^{2s_j^{(i)}/T}} \cdot \sigma_j^2. \tag{5}$$

Inserting Eq. (3) into Eq. (2):

$$\tilde{z}^{(i)} - z^{(i)} \;\approx\; -\sum_j w_j \,\frac{z^{(i)}}{p_j^{*(i)}}\,\epsilon_j^{(i)}. \tag{6}$$

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

$$\mathrm{Var}_\epsilon\big[\tilde{z}^{(i)}\big] = \sum_j w_j^2 \,\tau_j^{2,(i)}. \tag{7}$$

The quantity Eq. (7) depends on i through $\tau_j^{2,(i)}$. Since we seek weights independent of i, we minimize the variance *averaged* over data points:

$$\begin{aligned} J(\mathbf{w}) \;&=\; \mathbb{E}_i\big[\,\mathrm{Var}_\epsilon\big[\tilde{z}^{(i)}\big]\big] \\ &=\; \mathbb{E}_i\Big[\sum_j w_j^2\,\tau_j^{2,(i)}\Big] \\ &=\; \sum_j w_j^2\;\mathbb{E}_i\big[\tau_j^{2,(i)}\big] \\ &\equiv\; \sum_j w_j^2\;\bar{\tau}_j^{\,2} \end{aligned} \tag{8}$$

Substituting Eq. (5):

$$\bar{\tau}_j^{\,2} \;=\; \sigma_j^2\;\mathbb{E}_i\left[\frac{\big(z^{(i)}\big)^4}{e^{2s_j^{(i)}/T}}\right], \tag{9}$$

We minimize Eq. (8) subject to $\sum_j w_j = 1$. The Lagrangian is

$$\mathcal{L} = \sum_j w_j^2\;\bar{\tau}_j^{\,2} - \lambda\left(\sum_j w_j - 1\right). \tag{10}$$

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:

$$\sum_j \frac{\lambda}{2\bar{\tau}_j^{\,2}} = 1 \quad\Longrightarrow\quad \lambda = \frac{2}{\displaystyle\sum_k 1/\bar{\tau}_k^{\,2}}. \tag{11}$$

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

$$\boxed{w_j \;=\; \frac{1/\bar{\tau}_j^{\,2}}{\displaystyle\sum_k 1/\bar{\tau}_k^{\,2}} \;\propto\; \frac{1}{\bar{\tau}_j^{\,2}} \;=\; \frac{1}{\sigma_j^2\;\mathbb{E}_i\left[\dfrac{(z^{(i)})^4}{e^{2s_j^{(i)}/T}}\right]}}. \tag{12}$$

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:

$$S_j^2 \;\equiv\; \mathrm{Var}_i\big[\hat{p}_j^{(i)}\big], \qquad R_j^2 \;\equiv\; \mathrm{Corr}_i\big[p_j^{*(i)},\, \hat{p}_j^{(i)}\big]^2. \tag{13}$$

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:

$$S_j^2 = \mathrm{Var}_i\big[p_j^{*(i)} + \epsilon_j^{(i)}\big] = \mathrm{Var}_i\big[p_j^{*(i)}\big] + \sigma_j^2. \tag{14}$$

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:

$$R_j^2 = \frac{\mathrm{Cov}_i[p_j^*,\ \hat{p}_j]^2}{\mathrm{Var}_i[p_j^*] \cdot S_j^2} = \frac{\mathrm{Var}_i[p_j^*]^2}{\mathrm{Var}_i[p_j^*] \cdot S_j^2} = \frac{\mathrm{Var}_i[p_j^*]}{S_j^2}. \quad (15)$$

Substituting $\mathrm{Var}_i[p_j^{*(i)}] = R_j^2\, S_j^2$ into Eq. (14):

$$\boxed{\sigma_j^2 \ = \ S_j^2\,(1 - R_j^2).} \quad (16)$$

II. Effect of smoothing on score recovery

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 Table I 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 (Table II): applying smoothing to LR and kNN, which predict absolute scores rather than relative probabilities, degrades their performance.

TABLE I: Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset with and without tag-based smoothing for SCOUT.

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 II: Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset with and without tag-based smoothing applied to the kNN and LR baselines.

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	0.6611 ± 0.0035	0.4710 ± 0.0022	0.5178 ± 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:

$$\tilde{z}_j^{(i)} \ = \ \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}, \quad (17)$$

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

$$T \cdot \ln\big(\hat{p}_j^{(i)} \cdot \tilde{z}_j^{(i)}\big) \ = \ T \cdot \ln\left(\hat{p}_j^{(i)} \cdot \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}\right) \ = \ s_j^{(i)}. \quad (18)$$

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.

III. 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 (Table III).

TABLE III: 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) [1], ResNet-50 (2048-d) [2], ViT-B/16 (768-d) [3], and ConvNeXt-B (1024-d) [4], 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 [5] and YCB [6] 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 [7]. Examples of degenerate versus nondegenerate views are shown in Fig. 1 and examples of different lighting styles are shown in Fig. 2. 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.

Fig. 1: 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 [7].

Fig. 2: 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 [8] (version 2.0 throughout), InstantMesh [9], TRELLIS [10], and TripoSR [11], using default parameters for all models. For Hunyuan3D, we applied a convex decomposition using CoACD [12] (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.

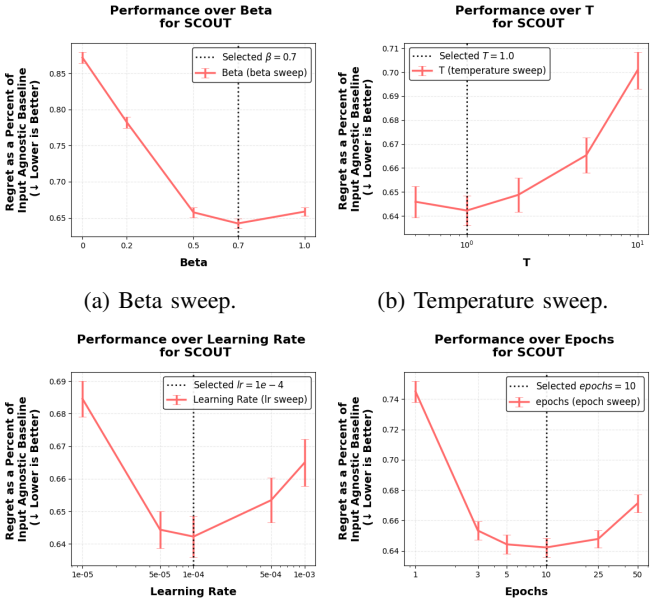
We also generated a mesh for three of the four view-invariant models: 2DGS [13], Nerf2Mesh [14], and NeuS2 [15]. We did not need to do any reconstruction for the fourth view-invariant model, the option to *skip*, as this has a fixed default score. For each of these three methods, we used 120 images to produce a mesh. Images were captured at elevation angles of $35°, 45°, 60°, 75°$, and $85°$. All elevations were captured at 24 azimuth angles spanning $360°$. There was minimal difference in reconstruction quality between using 120 images versus smaller numbers such as 24. We used the default parameters for each pipeline.

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* [16]. We then registered each reconstructed mesh to its ground-truth counterpart via FoundationPose [17] for global registration (default parameters), followed by DDM [18] for local refinement (learning rate 2e-2, 100 iterations).

Cost vectors. The AIQ and deferral curve metrics require latency, memory, and latency⊙memory cost vectors. Latency costs are 30s (Hunyuan3D), 10s (InstantMesh), 10s (TRELLIS), 0.5s (TripoSR), 10.9min (2DGS), 10.9min (Nerf2Mesh), and 5.5min (NeuS2). Memory costs are 6GB (Hunyuan3D), 24GB (InstantMesh), 16GB (TRELLIS), and 6GB (TripoSR). We exclude view-invariant methods from the memory cost vector because those models would dominate the Image-to-3D models under memory and latency⊙memory, and we exclude the *skip* option from all three cost families. Importantly, the routing models are trained on the full model pool regardless of which subset is used for evaluation. All values reflect mesh generation only (excluding texturing) and are based on official documentation [19], [20], [21], [22] when available, otherwise approximated by experiments.

IV. 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. 3. 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. 4).



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

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

Fig. 3: Sensitivity of hyperparameters for SCOUT evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

We repeat this analysis for SCOUT without decoupling, using 25 epochs instead of 10, and find a similarly smooth

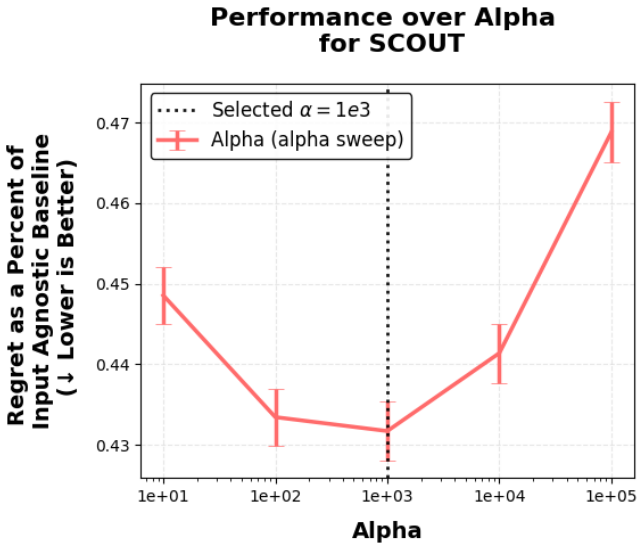
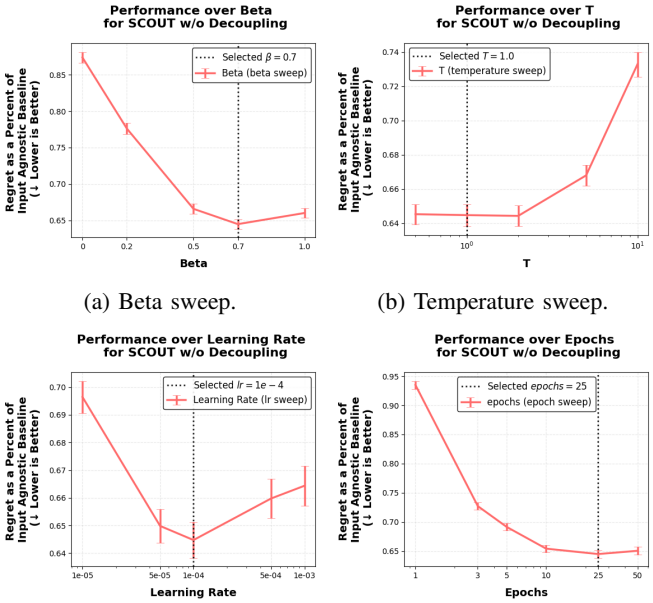


Fig. 4: Sensitivity of alpha for SCOUT evaluated on novel objects in the GSO dataset over cost coefficient vectors sampled from $\mathcal{C}$.



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

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

Fig. 5: Sensitivity of hyperparameters for SCOUT without decoupling evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

landscape with the same remaining hyperparameters (Fig. 5). 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 [23] 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. 6; LR: Fig. 7).

V. Additional Results

A. 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 (Table IV). 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 true probability is frequently low, making the proxy $\tilde{z}^{(i)}_j$ a higher-variance estimate of $z^{(i)}$ (see Eq. (4)).

B. GSO and YCB Results

Here we present more comprehensive tables of the GSO and YCB results. Table V reports regret across cost subspaces on GSO using DCD and Table VI reports regret on BigBIRD + YCB across seven quality metrics. All regrets are for novel objects and we evaluate on DCD for the GSO dataset and seven mesh quality metrics for the YCB dataset. Different metrics induce different optimal routing policies: IoU is volume-based and penalizes hollow geometry, whereas DCD measures surface-level correspondences. Since different manipulation tasks may require different quality criteria, a routing framework must be robust across metrics.

For each dataset, view-invariant score predictions are produced by one of two strategies: regressing on image embeddings, or using the tag category mean of view-invariant scores. The better-performing strategy is selected per metric based on validation performance. We find that the better choice depends on the noise characteristics of the dataset: the regression approach wins on GSO, while the tag-based mean wins in the majority of the BigBIRD + YCB settings (due to the smaller dataset size and more noise).

Comparison to baselines. For the GSO objects, SCOUT achieves statistically significant lower regret than all baselines in three of four cost subspaces, and is within the error bar of the kNN in the fourth. For the YCB objects, SCOUT achieves the lowest regret on all metrics except CD, where the input-agnostic baseline performs the best. This is due to the view-invariant methods often producing meshes with interior artifacts and irregular surfaces which yield noisy CD; all learned routers including ours are affected. Overall, we find that the closest baselines to SCOUT are the kNN and LR. Not only does SCOUT outperform both, achieving under half the regret of both kNN and LR in some experiments, but SCOUT is also more efficient and scalable than the kNN and LR baselines. In our experiments, we found SCOUT is $8.84\times$ faster than kNN (minutes vs. seconds) and $1.18\times$ faster than LR. The gap grows with the number of viewpoint-dependent models: SCOUT's runtime is effectively independent of k_1, whereas kNN and LR both scale linearly, an advantage that

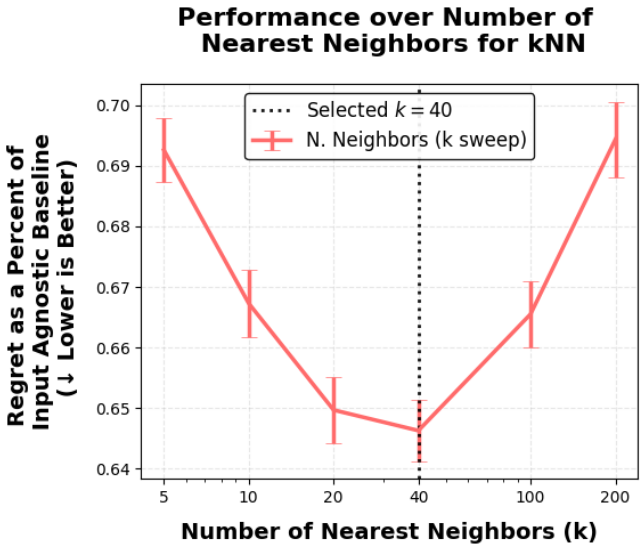


Fig. 6: Sensitivity of the number of nearest neighbors for the kNN baseline evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

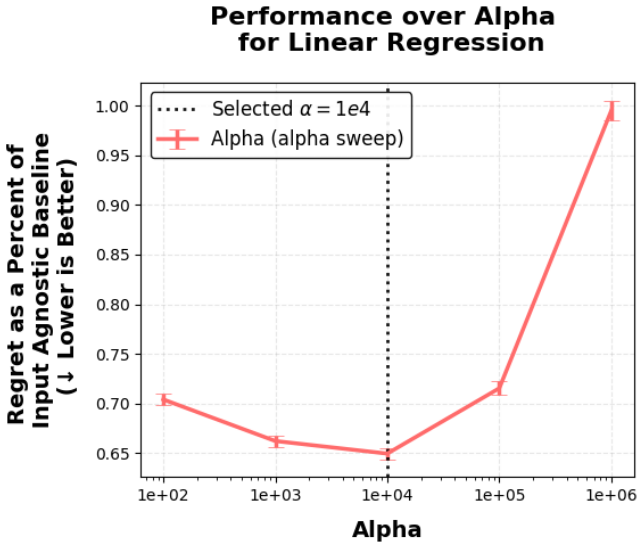


Fig. 7: Sensitivity of alpha for the linear regression baseline evaluated on novel objects in the GSO dataset over the cost coefficient vector $\mathcal{C}_0$.

increases as new Image-to-3D models emerge.

C. One dimensional cost vector families.

As the LLM routing problem originated from selecting between paid API endpoints, cost is expressed in dollars per API call. Therefore, Routerbench [24] introduces metrics for evaluating a routing framework along a single cost dimension $c(m)$ with scalar tradeoff λ:

$$m^* = \arg\max_{m \in \mathcal{M}} s(x, m) - \lambda \cdot c(m). \tag{19}$$

They note that 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. 8. They then introduce the Zero router baseline, which randomly samples routers at a given total cost allocation to trace out the non-decreasing convex hull as well as the Average Improvement in Quality (AIQ) defined as:

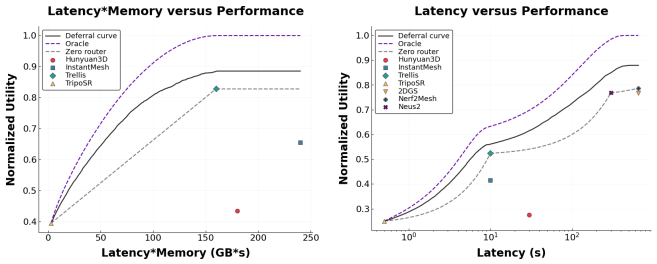
$$\mathrm{AIQ}(h) = \frac{1}{c_{max} - c_{min}} \int_{c_{min}}^{c_{max}} R_h \, \mathrm{d}c$$

We use these metrics for evaluating on the following 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.

ZOOTER and RouterDC are excluded from this evaluation as they do not support varying λ. SCOUT achieves the highest AIQ for all three cost families (Table VII). The magnitude of improvement over baselines is consistent with the inter-method differences reported in RouterBench [24], where gains between routing methods are similarly modest.

TABLE IV: 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 $\mathcal{C}$. 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



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

Fig. 8: Deferral curves for the latency⊙memory (a) and latency (b) cost coefficient vectors, showing the normalized utility achieved by each routing method as a function of the imposed cost constraint.

TABLE V: Regret (↓) as a proportion of the input-agnostic baseline regret across different cost subspaces on the GSO dataset. Ours (no dc) refers to SCOUT without decoupling.

Method	$\{\mathbf{c} \in \mathcal{C}\}$	$\{\mathbf{c} \in \mathcal{C} \cap \mathbf{c}_{\text{inv}} = \infty\}$	$\{\mathbf{c} = \mathbf{0}\}$	$\{\mathbf{c}_{\text{dep}} = \mathbf{0} \cap \mathbf{c}_{\text{inv}} = \infty\}$
ZOOTER	N/A	N/A	N/A	0.6769 ± 0.0039
RouterDC	N/A	N/A	N/A	0.8013 ± 0.0050
MLP	0.7993 ± 0.0035	0.5819 ± 0.0025	0.6626 ± 0.0050	0.7319 ± 0.0045
MF	0.6870 ± 0.0055	0.5667 ± 0.0035	0.7356 ± 0.0052	0.7584 ± 0.0080
kNN	0.4917 ± 0.0021	0.4992 ± 0.0019	$\mathbf{0.6261 \pm 0.0049}$	0.6489 ± 0.0033
LR	0.4909 ± 0.0019	0.5046 ± 0.0019	0.6361 ± 0.0046	0.6497 ± 0.0033
Ours (no dc)	0.5249 ± 0.0021	0.5146 ± 0.0019	0.6580 ± 0.0050	0.6769 ± 0.0039
Ours	$\mathbf{0.4734 \pm 0.0019}$	$\mathbf{0.4810 \pm 0.0019}$	0.6340 ± 0.0046	$\mathbf{0.6282 \pm 0.0035}$
Input-agnostic	1.0000	1.0000	1.0000	1.0000

TABLE VI: 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	1.0303 ± 0.0164	15.2150 ± 2.7513	1.7054 ± 0.1696	0.7681 ± 0.0166	1.0942 ± 0.0513	0.8845 ± 0.0372	0.8905 ± 0.0132
MF	1.2078 ± 0.0175	13.1332 ± 1.6261	2.2107 ± 0.1350	0.9076 ± 0.0148	1.9530 ± 0.0570	2.7425 ± 0.1688	1.3766 ± 0.0317
kNN	0.9282 ± 0.0166	1.7599 ± 0.2032	1.0074 ± 0.0776	0.7474 ± 0.0143	0.9788 ± 0.0437	1.1768 ± 0.1274	0.8522 ± 0.0184
LR	0.8919 ± 0.0168	7.8104 ± 1.2405	1.4418 ± 0.1284	0.7330 ± 0.0142	1.0073 ± 0.0493	0.7511 ± 0.0383	0.8258 ± 0.0211
Ours (no dc)	0.9258 ± 0.0154	10.8840 ± 1.9559	1.3402 ± 0.1385	0.6700 ± 0.0131	0.9678 ± 0.0500	0.6893 ± 0.0570	0.8576 ± 0.0289
Ours	$\mathbf{0.8417 \pm 0.0165}$	6.6110 ± 0.9944	1.3728 ± 0.1610	$\mathbf{0.6248 \pm 0.0147}$	$\mathbf{0.9146 \pm 0.0356}$	$\mathbf{0.3547 \pm 0.0109}$	$\mathbf{0.8243 \pm 0.0223}$
Input-agnostic	1.0000	**1.0000**	**1.0000**	1.0000	1.0000	1.0000	1.0000

TABLE VII: AIQ (↑) for one-dimensional cost families on the GSO dataset, evaluated on novel objects.

Method	**Memory**	**Latency⊙Memory**	**Latency**
MLP	0.8158 ± 0.0007	0.7756 ± 0.0007	0.5730 ± 0.0010
MF	0.8138 ± 0.0012	0.7722 ± 0.0012	0.5661 ± 0.0012
kNN	0.8307 ± 0.0006	0.7847 ± 0.0007	0.5772 ± 0.0010
LR	0.8284 ± 0.0006	0.7832 ± 0.0007	0.5774 ± 0.0010
Ours (no dc)	0.8275 ± 0.0006	0.7826 ± 0.0007	0.5763 ± 0.0010
Ours	$\mathbf{0.8348 \pm 0.0006}$	$\mathbf{0.7887 \pm 0.0007}$	$\mathbf{0.5801 \pm 0.0010}$
Zero router	0.7126 ± 0.0008	0.6914 ± 0.0006	0.5539 ± 0.0008
Oracle	0.9459 ± 0.0003	0.8813 ± 0.0005	0.6639 ± 0.0009

D. Grasp collisions

Figure 9 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 10 and 11 show the eight remaining objects from the grasp collision experiment. 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.

E. Policy learned

To understand what SCOUT learns, we examine which model is selected as a function of object category (Fig. 12) and input viewpoint (Fig. 13). 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.

REFERENCES

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PmLR, 2021.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

[5] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2553–2560, Ieee, 2022.

[6] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*, pp. 510–517, IEEE, 2015.

[7] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 509–516, IEEE, 2014.

[8] T. H. Team, "Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation," 2025.

[9] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan, "Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models," 2024.

[10] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang, "Structured 3d latents for scalable and versatile 3d generation," *arXiv preprint arXiv:2412.01506*, 2024.

[11] D. Tochilkin, D. Pankratz, Z. Liu, Z. Huang, A. Letts, Y. Li, D. Liang, C. Laforte, V. Jampani, and Y.-P. Cao, "Triposr: Fast 3d object reconstruction from a single image," 2024.

[12] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[13] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

[14] J. Tang, H. Zhou, X. Chen, T. Hu, E. Ding, J. Wang, and G. Zeng, "Delicate textured mesh recovery from nerf via adaptive surface refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17739–17749, 2023.

[15] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu, "Neus2: Fast learning of neural implicit surfaces for multi-

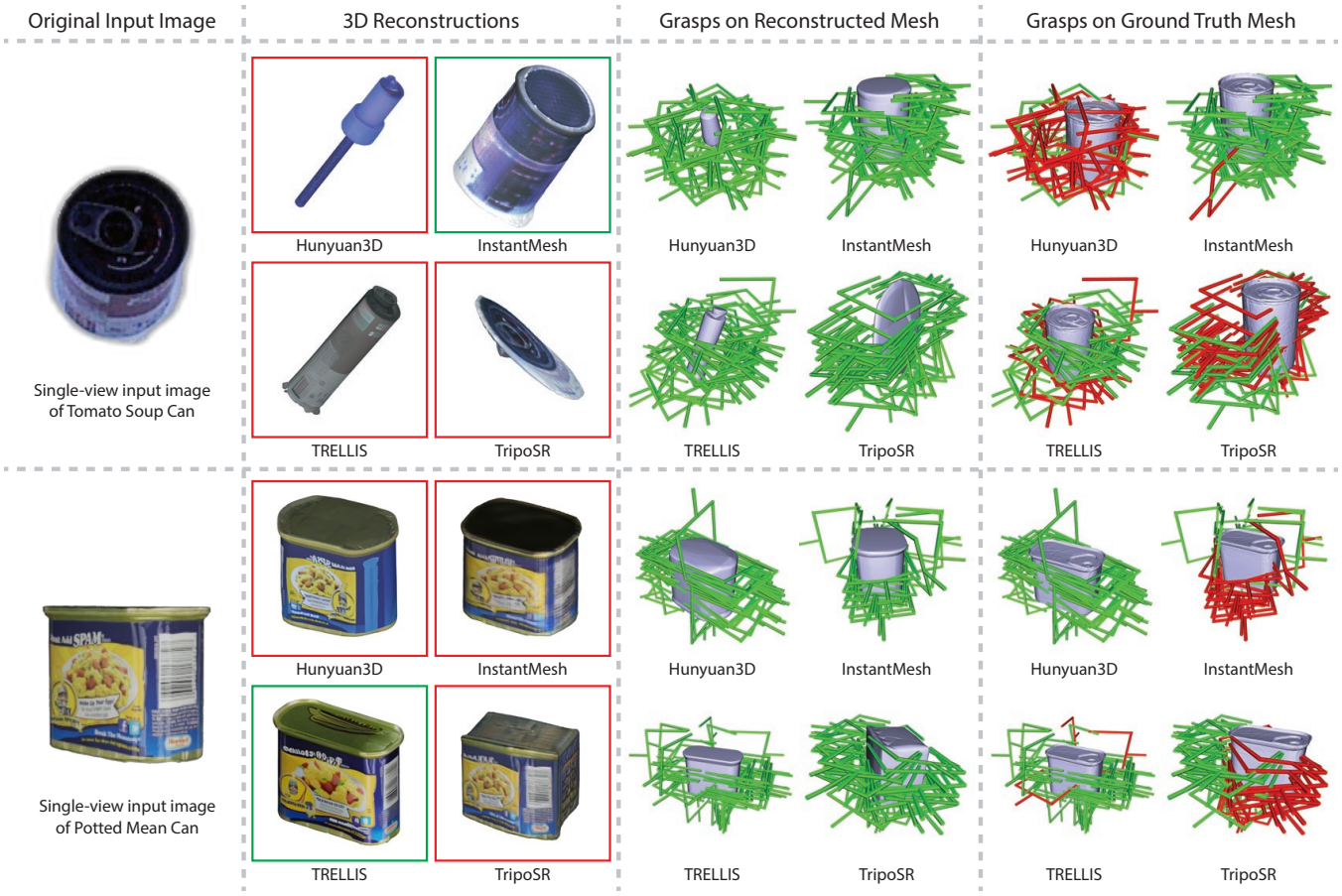


Fig. 9: Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red.

view reconstruction," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3295–3306, 2023.

[16] A. 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.

[17] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17868–17879, 2024.

[18] S. Ren, J. Hou, X. Chen, H. Xiong, and W. Wang, "Ddm: A metric for comparing 3d shapes using directional distance fields," *arXiv preprint arXiv:2401.09736*, 2024.

[19] Tencent Hunyuan Team, "Hunyuan3D-2 source code repository." `https://github.com/Tencent-Hunyuan/Hunyuan3D-2`, 2025.

[20] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan, "InstantMesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models." `https://github.com/TencentARC/InstantMesh`, 2024.

[21] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang, "TRELLIS: Structured 3d latents for scalable and versatile 3d generation." `https://github.com/microsoft/TRELLIS`, 2024.

[22] D. Tochilkin, D. Pankratz, Z. Liu, Z. Huang, A. Letts, Y. Li, D. Liang, C. Laforte, V. Jampani, and Y.-P. Cao, "TripoSR: Fast 3d object reconstruction from a single image." `https://github.com/VAST-AI-Research/TripoSR`, 2024.

[23] Y. Li, "Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers," *arXiv preprint arXiv:2505.12601*, 2025.

[24] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, and S. K. Upadhyay, "Routerbench: A benchmark for multi-llm routing system," *arXiv preprint arXiv:2403.12031*, 2024.

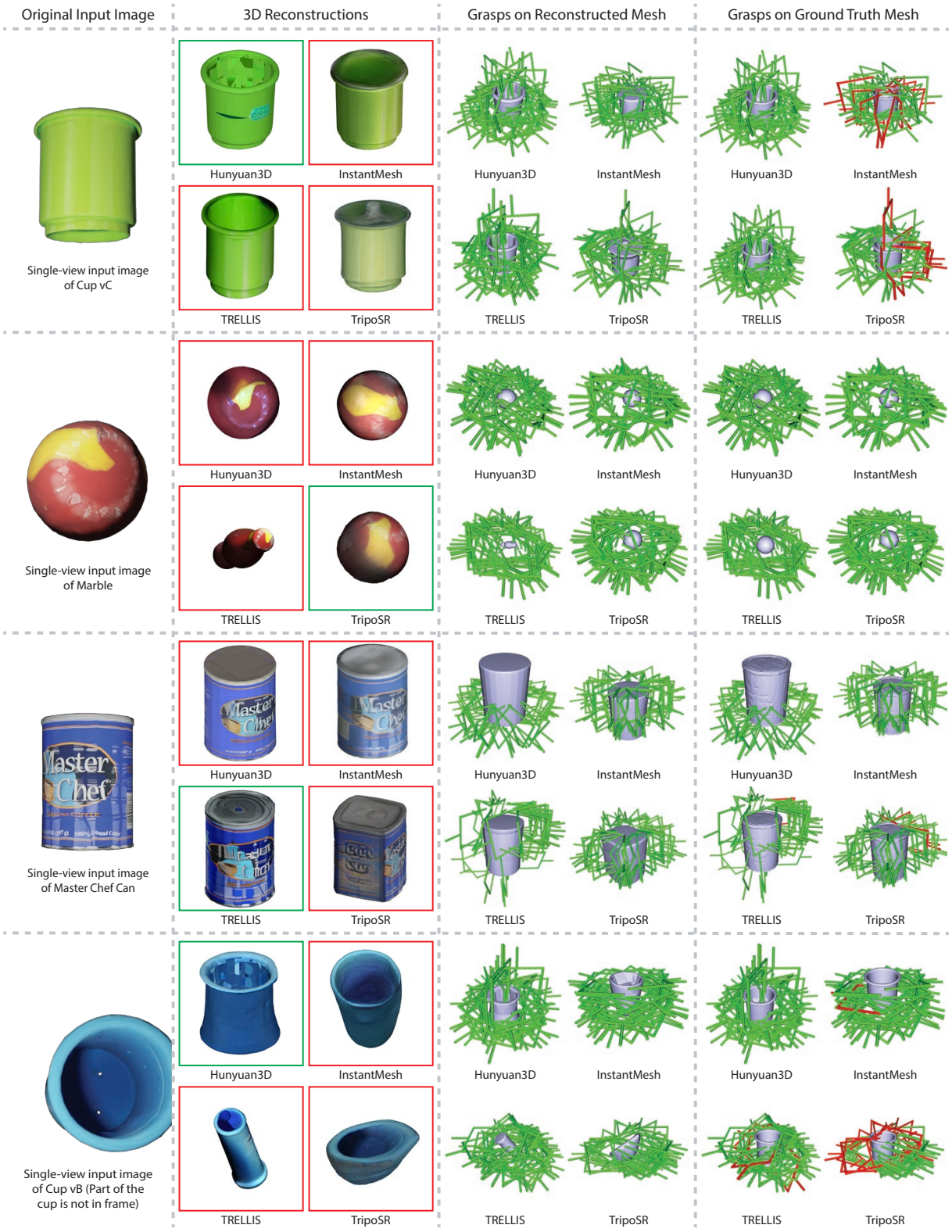


Fig. 10: Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red.

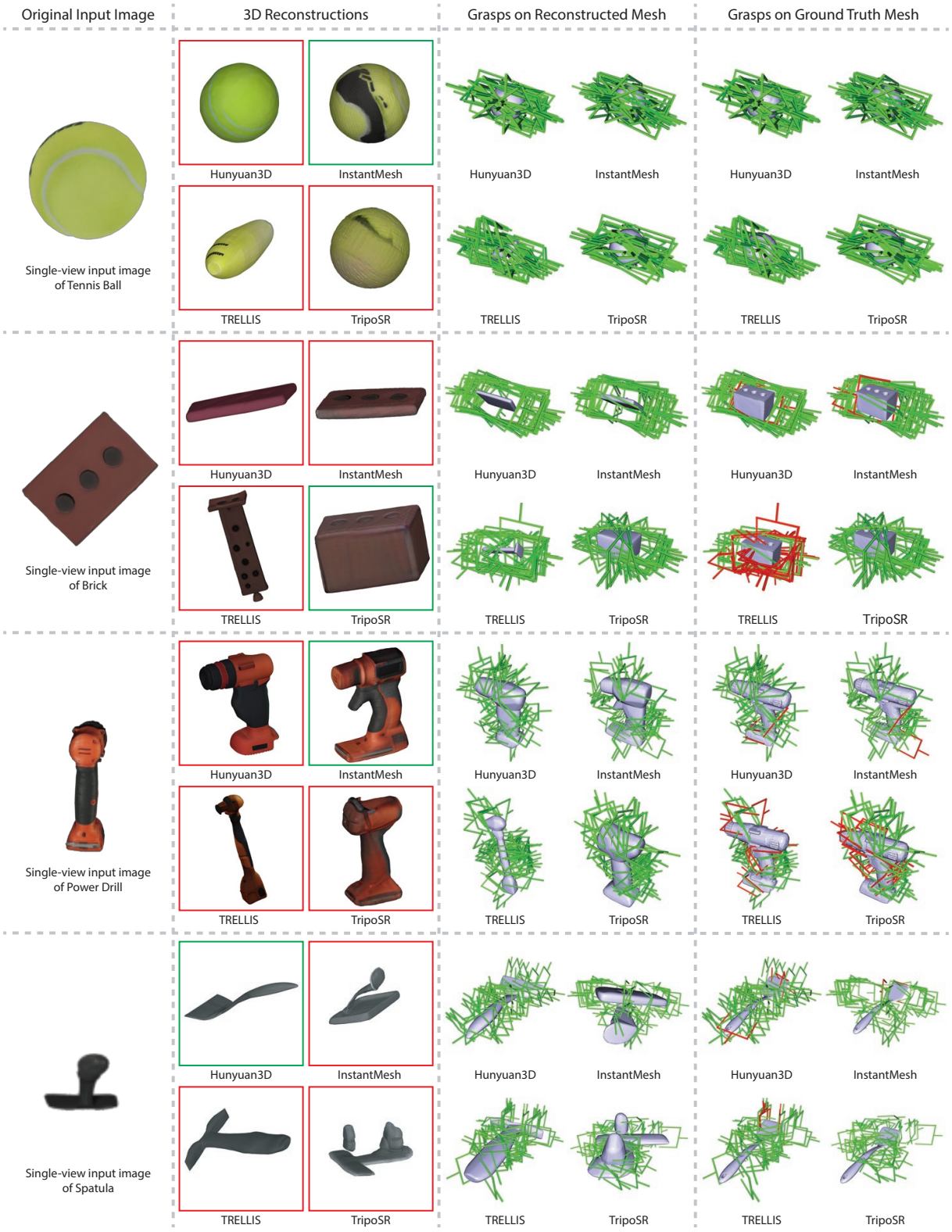
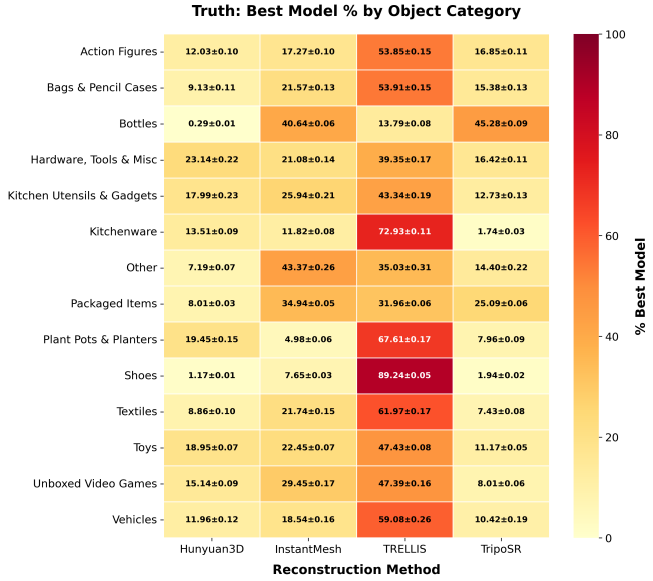
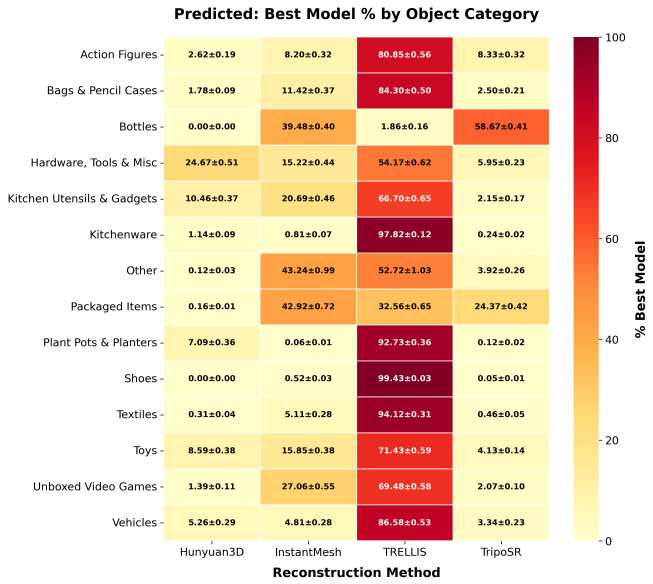


Fig. 11: Evaluation of grasp proposals based on the reconstructed meshes from the input image. The figure shows the following from left to right: (1) original input image inputted to SCOUT, (2) 3D reconstructions from the original image with the model chosen by SCOUT boxed in green, (3) grasp proposals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

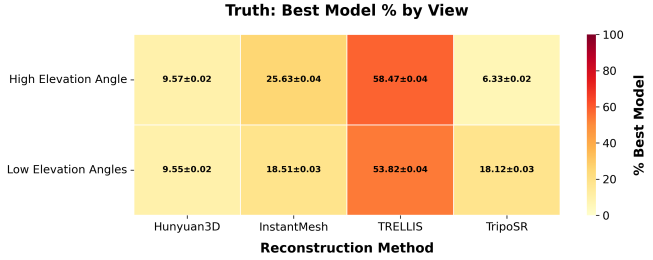


(a) Oracle policy: how often each model is truly optimal when evaluated on unseen objects, grouped by object category.

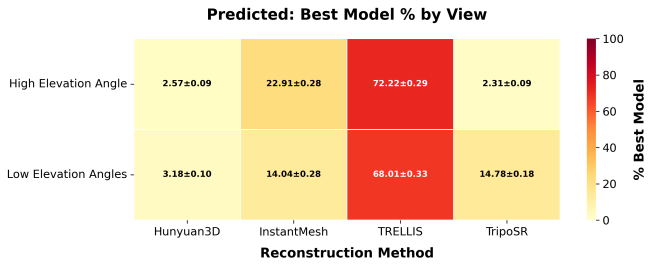


(b) SCOUT's learned policy: how often each model is selected when evaluated on unseen objects, grouped by object category.

Fig. 12: 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.

Fig. 13: Oracle and learned routing policies across viewpoint elevations. High: 75° and 85°; low: 35°, 45°, and 60°.