

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

by

Akash K. Anand

S.B. Computer Science and Engineering, Massachusetts Institute of Technology, 2026.

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2026

© 2026 Akash K. Anand. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Akash K. Anand
Department of Electrical Engineering and Computer Science
May 15, 2026

Certified by: Leslie Pack Kaelbling
Panasonic Professor of Computer Science and Engineering, Thesis Supervisor

Certified by: Aditya Agarwal
Ph.D. Student, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

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

by

Akash K. Anand

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2026 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE

ABSTRACT

Robotic manipulation tasks impose varying quality requirements on 3D reconstruction: 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 that aggregate information across multiple views rather than relying on the initial robot viewpoint. Querying all models is computationally prohibitive, which motivates 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, represented by a learned probability distribution, and (2) the overall image difficulty, estimated by a scalar partition function. SCOUT supports arbitrary cost functions at inference time, accommodating the multi-dimensional cost constraints that are common in robotics. We evaluate SCOUT 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 regimes. We further validate the framework through robotic grasping and dexterous manipulation experiments. We release the code and additional results on our [website](#).

Thesis supervisor: Leslie Pack Kaelbling

Title: Panasonic Professor of Computer Science and Engineering

Thesis supervisor: Aditya Agarwal

Title: Ph.D. Student

Acknowledgments

First, I would like to thank my thesis supervisors. I am incredibly grateful to Aditya for teaching me so much and making the lab such an enjoyable place to work. I am also deeply grateful to Leslie, who has shaped the way I think about robots and problem solving through her classes, my undergraduate research, and now this thesis. I would also like to thank all the friends I have made at MIT over these past four years for all the fun we have had together. Finally, I am so grateful to my parents, brothers, and grandparents for all the support they have given me.

Contents

<i>List of Figures</i>	9
<i>List of Tables</i>	11
1 Introduction	13
2 Related Work	17
2.1 Image-to-3D Reconstruction	17
2.2 Multi-view Reconstruction	17
2.3 Model Routing for Large Language Models	19
2.4 Reconstruction-Based Manipulation Pipelines	20
3 Problem Formulation	21
3.1 Setup	21
3.2 Score and Cost functions	22
3.3 Routing objective	22
3.4 Comparison with LLM Routing	23
4 SCOUT	25
4.1 Preprocessing	25
4.2 Score Decoupling	26
4.3 Utility Optimization	28
4.4 Derivation of Optimal Weights	28
4.4.1 Assumptions	29
4.4.2 Global weight optimization	30
4.4.3 Estimation of σ_j^2	31
5 Experiments	33
5.1 Experimental Setup	33
5.1.1 Mesh Reconstruction Models	33
5.1.2 Datasets	33
5.1.3 Mesh Quality Metrics	34
5.1.4 Predicting Scores for View-Invariant Methods	34
5.1.5 Cost Families for Evaluations	35
5.1.6 Baselines	37
5.1.7 Evaluation Metrics	38
5.1.8 Implementation Details	38

5.2	Routing Performance on Benchmark Datasets	40
5.2.1	Regret Across Cost Families on GSO	41
5.2.2	Regret Across Quality Metrics on BigBIRD + YCB	41
5.2.3	One-Dimensional Cost Families	42
5.2.4	Comparison with Adapted LLM Routing Baselines	43
5.3	Robotic Manipulation Validation	43
5.3.1	Grasp Proposal Evaluation	44
5.3.2	Dexterous Manipulation in Simulation	45
5.3.3	Real-World Pick-and-Place on a Franka Panda	45
5.4	Design Choice Ablations	47
5.4.1	Effect of Smoothing	47
5.4.2	Weighted Proxy Comparison	48
5.4.3	Hyperparameter Sensitivity	49
5.5	What SCOUT Learns	50
5.5.1	Per-Category Routing Behavior	50
5.5.2	Per-Viewpoint Routing Behavior	50
5.5.3	Implications	51
6	Conclusion	57
6.1	Summary	57
6.2	Future Work	58
6.3	Impact	59
	<i>References</i>	61

List of Figures

1.1	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.	13
1.2	Architecture of SCOUT. SCOUT takes as input the current input image and user-specified costs per model, decouples viewpoint-dependent and view-invariant scores, and selects a reconstruction pipeline to build the output mesh.	15
2.1	Example of different biases of Image-to-3D models. TRELLIS adds long artifacts on a top-down view of an apple but produces an accurate mesh on the real image. Hunyuan3D adds artifacts due to the imperfect background removal of real images but produces an accurate mesh on the top-down view.	18
4.1	InstantMesh and TripoSR (viewpoint-dependent) produce high-quality reconstructions on the easy input (top) and degraded reconstructions on the hard input (bottom)—their scores share a common difficulty-induced offset. NeuS2 (view-invariant) captures its own views during reconstruction, so the quality of its produced mesh is uncorrelated with the input image’s difficulty. A single softmax over all three models would conflate these two regimes; decoupling separates them.	27
4.2	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.	29
5.1	Degenerate and nondegenerate views of a cracker box.	39
5.2	Different lighting conditions for a fixed viewpoint.	40
5.3	Deferral curves for the three cost families $\mathcal{C}_{\text{latency}}$ (left), $\mathcal{C}_{\text{memory}}$ (middle), and $\mathcal{C}_{\text{latency} \times \text{memory}}$ (right), showing the normalized utility achieved by each routing method as a function of the imposed cost constraint.	43

5.4	Evaluation of grasp proposals on reconstructed meshes for two example inputs. From left to right: (1) original input image to SCOUT, (2) 3D reconstructions from each model 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. SCOUT selects InstantMesh for the top-down can view (where it is the only model producing a usable reconstruction) and TRELLIS for the frontal meat container view (where it reconstructs accurately at lower latency than Hunyuan3D).	46
5.5	Evaluation of grasp proposals on the first half of the remaining eight YCB objects. Layout follows Figure 5.4.	52
5.6	Evaluation of grasp proposals on the second half of the remaining eight YCB objects. Layout follows Figure 5.4.	53
5.7	Sensitivity of SCOUT to the three hyperparameters affecting the relative-performance model $\hat{\mathbf{p}}$, evaluated on novel GSO objects under the cost function $c_{\text{view, no-cost}}$. Each curve exhibits a clear local optimum and smooth degradation away from it.	54
5.8	Sensitivity of SCOUT to the two hyperparameters affecting the partition function model $\hat{z}$, evaluated on novel GSO objects under the cost-function family $\mathcal{C}$. Each curve exhibits a clear local optimum and smooth degradation away from it.	54
5.9	Sensitivity of the two strongest baselines to their primary hyperparameters, evaluated on novel GSO objects under the cost function family $\mathcal{C}$	54
5.10	Oracle and learned routing policies across object categories. The learned policy closely tracks the oracle’s category-dependent preferences, indicating that SCOUT generalizes the category structure to novel objects rather than memorizing per-object selections.	55
5.11	Oracle and learned routing policies across viewpoint elevations. High: 75° and 85°; low: 35°, 45°, and 60°. The learned policy reproduces the oracle’s viewpoint-dependent structure, confirming that SCOUT captures viewpoint-conditional model preferences and applies them to novel objects.	56

List of Tables

5.1	Within-category mean absolute deviation (MAD) of scores for the two viewpoint-handling groups. For each metric, we group (model, view) pairs by object category, compute the MAD around the per-category mean within each category, weight-average across categories by category size, and average across models in each group. A ratio > 1 (bold) indicates that view-dependent scores are more spread within category than view-invariant scores.	36
5.2	Softmax temperature T per metric for BigBIRD + YCB.	38
5.3	Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret across different cost regimes on the GSO dataset, evaluated by DCD. SCOUT no dc refers to SCOUT (ours) without decoupling.	41
5.4	Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret for $c \in \mathcal{C}$ across multiple quality metrics on BigBIRD + YCB. SCOUT no dc denotes SCOUT (ours) without decoupling.	42
5.5	AIQ ($\uparrow$) for one-dimensional cost families on the GSO dataset, evaluated on novel objects.	42
5.6	Grasp collision rate ($\downarrow$) / mesh-IoU ($\uparrow$) across 10 YCB objects, using a two-fingered gripper. SCOUT is evaluated under two settings: routing over novel views of known objects, and over novel views and novel objects.	44
5.7	Dexterous manipulation success rate ($\uparrow$) in simulation for each reconstruction model across 5 objects (HY3D: Hunyuan3D, IM: InstantMesh, TRL: TRELIS, TSR: TripoSR). SCOUT (ours) reports the model selected by SCOUT for each object and the resulting mean success rate.	47
5.8	Real-world pick-and-place success rate ($\uparrow$) on a Franka Panda robot across 5 objects. For each reconstructed object, 5 antipodal grasps are sampled [59] and executed. SCOUT (ours) reports the model selected by SCOUT for each object and the resulting mean success rate.	47
5.9	Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret across the $c_{\text{view, no-cost}}$ regime on GSO, with and without tag-based smoothing. Lowest regret per method is in bold	48
5.10	Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret for $c \in \mathcal{C}$ across multiple quality metrics on BigBIRD + YCB. SCOUT uses the derived weighted average from Equation (4.1); True z uses the exact partition function; Equiweighted uses a uniform average across models; One-hot uses a single model. Lowest regret per metric is in bold	49

Chapter 1

Introduction

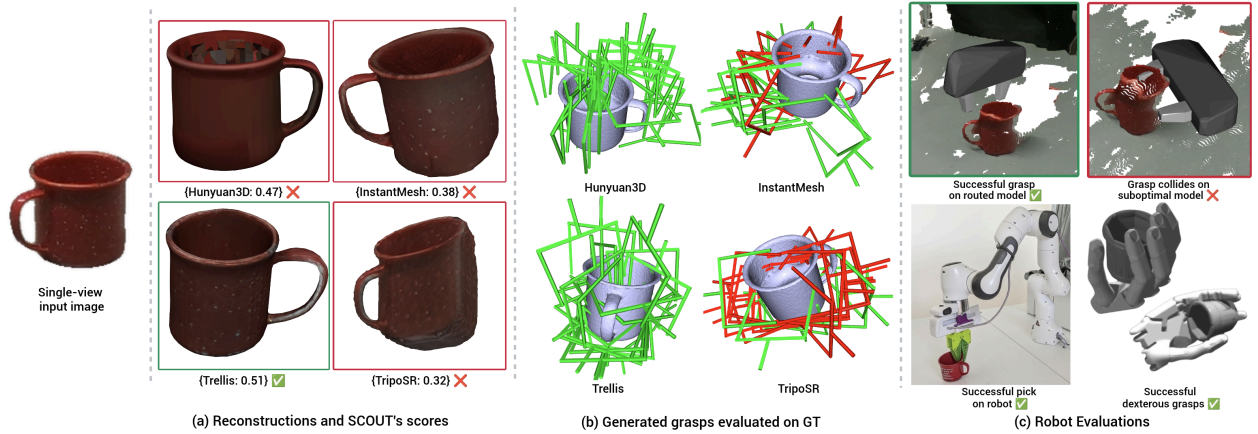


Figure 1.1: Overview of SCOUT. Given an input image, (a) SCOUT routes to the most suitable reconstruction model and generates the selected 3D reconstruction (compared against other candidate models). (b) Grasp proposals generated on the reconstructed mesh, evaluated on the ground-truth mesh; colliding grasps are shown in red. (c) Utility of SCOUT’s reconstruction-aware routing in downstream robust robot grasping and dexterous manipulation.

Single-image 3D reconstruction has advanced rapidly in recent years. Early optimization-based approaches demonstrated that strong generative priors [1,2] could lift a single image into a 3D representation, but required computationally expensive per-object optimization procedures [3–6] that often take minutes to hours to complete, limiting their applicability in real-time robotics applications. Recent Image-to-3D models have largely removed this bottleneck, producing mesh reconstructions within seconds and without per-object optimization. These models span a diverse range of architectures, including feed-forward prediction [7,8], multi-view diffusion followed by reconstruction [9,10], and iterative generation within learned 3D latent spaces [11,12], exhibiting complementary strengths. Some models prioritize inference speed, while others emphasize geometric fidelity or texture quality. Their relative performance varies across object categories and input viewpoints. As a result, no single approach dominates across all settings.

These advances are particularly significant for robotic manipulation, where a robot

often has access to only a single initial view of an object [13,14], and acquiring additional viewpoints may be time-consuming, operationally constrained, or altogether infeasible [15,16]. Several recent manipulation pipelines [13,17–20] have incorporated Image-to-3D models as perception modules, treating reconstruction as a building block for downstream planning and control. However, these systems typically commit to a fixed reconstruction model chosen a priori, which exposes the downstream pipeline to that model’s specific failure modes and forgoes the performance gains that may be obtained by choosing an Image-to-3D model per input. Furthermore, reconstruction quality requirements tend to vary substantially by task. Dexterous manipulation demands fine-grained surface detail, whereas collision-free motion planning can tolerate coarser representations. The choice of reconstruction method therefore affects both robustness and efficiency. Robustness captures how reliably the pipeline handles diverse inputs, while efficiency captures whether reconstruction meets the latency and resource budgets required for real-time deployment.

Beyond Image-to-3D methods, approaches such as Nerf2Mesh [21], NeuS2 [22], and 2DGS [23] take multiple images as input and produce reconstructions whose quality depends less strongly on the robot’s initial viewpoint and more heavily on factors such as object texture. We refer to these methods as *view-invariant*, in contrast to the *viewpoint-dependent* Image-to-3D models described above. While view-invariant methods can typically produce higher-quality reconstructions, they also require additional sensing time due to multi-capture requirements and longer optimization procedures.

A robotic system operating under tight latency or view-acquisition constraints may prefer a fast Image-to-3D model while accepting lower fidelity, whereas a system with an increased time budget and multi-view-friendly hardware — such as a wrist-mounted sensor that can capture many views [24,25] — may benefit from the superior quality provided by view-invariant reconstruction methods. The choice between these regimes — and among models within them — depends on the input viewpoint, task requirements, and available computational budget.

We propose a framework that learns to route each input to the most appropriate reconstruction model prior to inference. The routing decision balances reconstruction accuracy against user-specified costs such as latency, memory consumption, and monetary expenses. Although model routing has emerged as an active area of research in the context of Large Language Models (LLMs) [26–30], it has not previously been explored for 3D reconstruction. The reconstruction setting introduces several challenges that are not present in the LLM setting. First, the costs of reconstruction models are largely input-independent. Second, robotics applications must balance multiple cost dimensions, such as latency, memory, and energy, rather than optimizing against a single cost dimension such as API pricing used in LLM routing. Third, the model pool is structurally heterogeneous, encompassing both viewpoint-dependent methods (*e.g.*, Image-to-3D models) and view-invariant methods (*e.g.*, multi-view mesh reconstruction models that aggregate information across multiple views rather than relying on the initial robot viewpoint).

To address these challenges, we propose **SCOUT (Score-Conditioned Optimal Utility Targeting)**, an input-aware routing framework that decouples reconstruction scores into two components: (1) the relative performance of viewpoint-dependent models, represented by a learned probability distribution, and (2) the overall difficulty of the input image, estimated by a scalar partition function (overview of architecture shown in Figure 1.2). This decoupling

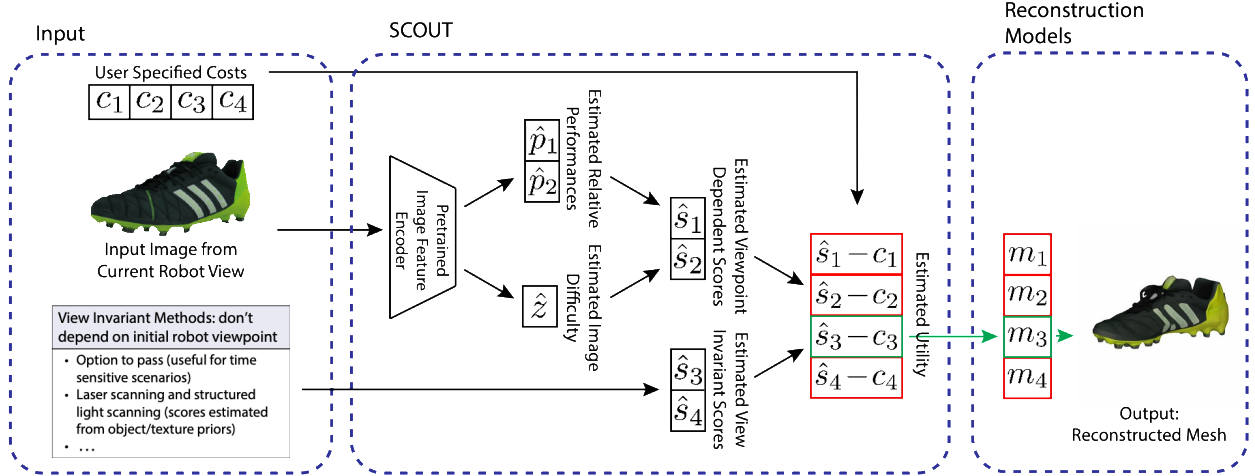


Figure 1.2: Architecture of SCOUT. SCOUT takes as input the current input image and user-specified costs per model, decouples viewpoint-dependent and view-invariant scores, and selects a reconstruction pipeline to build the output mesh.

stabilizes training and provides a key scalability advantage: the learned network predicts only over the viewpoint-dependent models, resulting in its size and training procedure being entirely independent of the view-invariant model pool. SCOUT supports arbitrary cost functions at inference time, giving a single trained router the ability to accommodate various deployment scenarios in which latency is paramount, memory is constrained, or both objectives are jointly optimized, all without retraining. Figure 1.1 illustrates the framework: given an input image, SCOUT selects the optimal reconstruction model under the user-specified cost function and then passes the resulting reconstruction output to downstream manipulation tasks such as grasping and dexterous manipulation.

We evaluate SCOUT on the Google Scanned Objects [31], BigBIRD [32], and YCB [33] datasets under multiple mesh quality metrics, including Density-aware Chamfer Distance (DCD) [34], IoU, and geometric metrics from Eval3D [35]. Across diverse cost-function regimes, SCOUT consistently outperforms routing baselines adapted from the LLM literature, achieving less than half the regret of the strongest competitors in some experimental settings. Beyond benchmark evaluation, we validate SCOUT’s 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. SCOUT routes to different reconstruction models depending on the input, empirically confirming that no single model dominates across the inputs encountered in deployment, and that per-input routing yields measurable improvements in downstream manipulation performance.

The contributions of this work are threefold. First, it introduces the first formulation of model routing for 3D reconstruction, accommodating both viewpoint-dependent and view-invariant methods under arbitrary cost functions. Second, it develops a theoretically-justified partition function proxy that recovers absolute scores from a learned relative distribution. Third, it proposes a decoupling procedure that separates image difficulty from relative model performance, which maintains independence between the learned router and the view-invariant

pool.

The remainder of this thesis is organized as follows. Chapter 2 reviews related work in Image-to-3D models, view-invariant reconstruction, model routing for LLMs, and reconstruction-based manipulation pipelines. Chapter 3 formalizes the model routing problem for 3D reconstruction and contrasts it with the LLM routing setting. Chapter 4 presents SCOUT, including the decoupling procedure and the partition function proxy, along with its derivation. Chapter 5 evaluates the proposed approach across benchmark datasets and robotic manipulation tasks. Chapter 6 concludes and outlines potential directions for future work.

Chapter 2

Related Work

2.1 Image-to-3D Reconstruction

Early single image 3D reconstruction methods relied on per-object optimization, adapting score distillation sampling [3,4] and image-conditioned novel view synthesis models [1] to iteratively lift a single image into a 3D representation [5,6]. Although these methods produced high-quality reconstructions, the optimization process frequently required minutes to hours per object, limiting their utility in real-time applications such as robotic manipulation. Subsequent work, which we collectively refer to as Image-to-3D models, eliminated per-object optimization [8,10,36], enabling the production of meshes from a single image in seconds. Among Image-to-3D models, four widely used pipelines exemplify the diversity of architectural design. **TripoS**R [7] prioritizes low-latency inference via a feed-forward transformer that produces a triplane-NeRF in under 0.5 seconds. **InstantMesh** [9] focuses on geometrically faithful reconstructions by pairing a multi-view diffusion model with sparse-view mesh reconstruction supervised by depth and normal losses. **Hunyuan3D** [12] emphasizes high-quality texture generation through a two-stage pipeline that decouples shape generation from texture synthesis. **TRELLIS** [11] employs a unified structured latent representation that decodes into multiple 3D formats, including radiance fields, Gaussians, and meshes. Collectively, these models cover a spectrum of design priorities: inference speed, geometric fidelity, texture quality, and representation flexibility. However, this leads to each model inheriting distinct biases from its training data and architecture. As a consequence, no single model dominates across all inputs encountered in deployment, with relative performance varying across object categories and viewpoints, as seen in Figure 2.1. This diversity creates a strong motivation for principled per-input model selection. Given a specific input and user-defined cost constraints, the ability to select the most appropriate reconstruction model allows for the exploitation of complementary model strengths while avoiding the limitations of fixed-model deployment.

2.2 Multi-view Reconstruction

A complementary line of work produces 3D reconstructions from multiple views rather than from a single image. These multi-view reconstruction methods often produce higher-

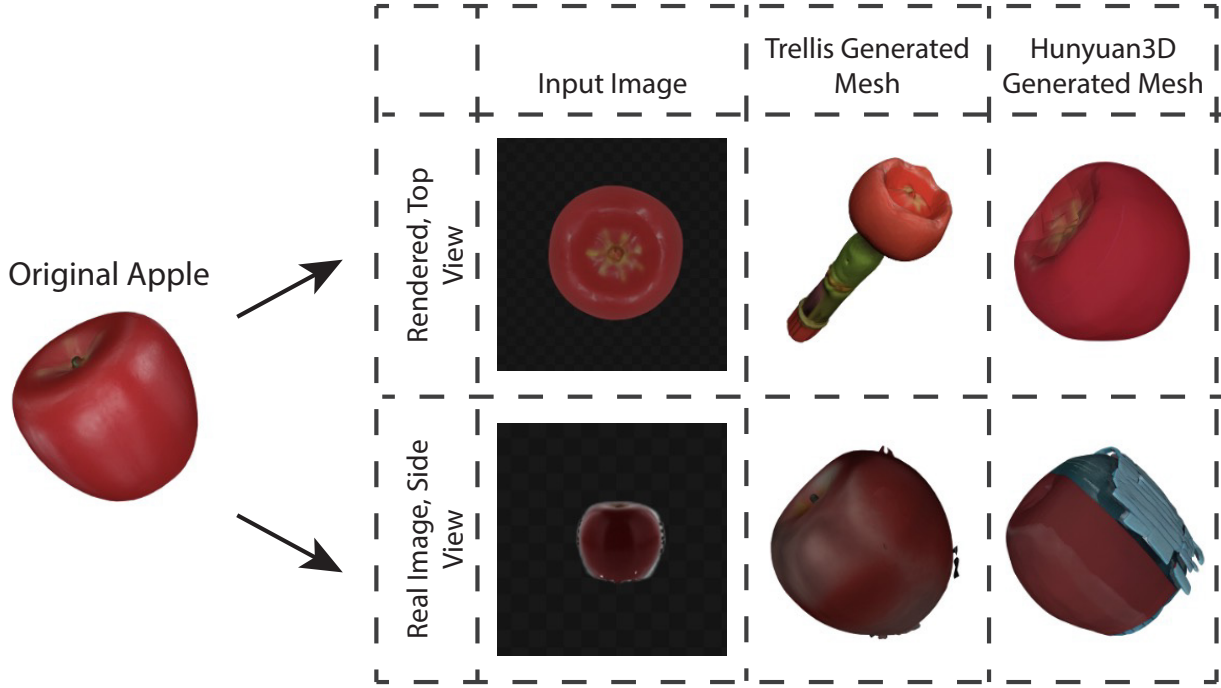


Figure 2.1: Example of different biases of Image-to-3D models. TRELLIS adds long artifacts on a top-down view of an apple but produces an accurate mesh on the real image. Hunyuan3D adds artifacts due to the imperfect background removal of real images but produces an accurate mesh on the top-down view.

quality outputs but incur additional sensing and computational costs due to multiple capture requirements, longer optimization, or both. They span several representational paradigms. NeRF [37] represents a scene as a continuous neural radiance field optimized from multi-view images, rather than producing meshes directly. **NeRF2Mesh** [21] extends this line by extracting meshes from a NeRF through iterative surface refinement. **NeuS** [38] represents the surface as the zero-level-set of a signed-distance field with a volume-rendering-compatible formulation, and **NeuS2** [22] accelerates this approach via multi-resolution hash encoding. 3D Gaussian Splatting [39] represents the scene as a set of anisotropic 3D Gaussians fit to multi-view images and rasterized through differentiable splatting. **2DGS** [23] adapts this to 2D Gaussian primitives that better approximate surfaces for mesh extraction. For the remainder of this thesis, we refer to these methods collectively as *view-invariant* reconstruction methods. Notably, these methods are not view-invariant in the strict sense, as they fundamentally depend on having access to multiple views, and the choice and number of views affect the output. However, the property we aim to focus on is invariance to the robot’s *initial view*. A robot starting from a face-on view and a robot starting from a corner view of the same object will obtain essentially the same reconstruction. This is because the reconstruction process moves beyond the initial viewpoint and captures additional views of the object. Given views of the object from various directions, the significance of the initial viewpoint with regard to the total information used in the reconstruction decreases significantly. This leads to outputs that are effectively independent of the robot’s initial viewpoint, at the cost of acquiring those

additional views. The cost of view acquisition varies considerably across deployments—a wrist-mounted depth sensor may capture additional views with minimal overhead, while a heavily occluded scene or kinematically constrained workspace may make additional captures expensive or infeasible. We do not explicitly model acquisition costs arising from occlusion or kinematic constraints in this work, and instead assume the user specifies a per-model cost that captures their deployment’s effective acquisition burden, potentially an infinite cost indicating that we cannot use multi-view reconstruction methods for a given problem. Further, we observe that both single-image [17,19] and multi-image [40,41] pipelines exist for mesh reconstruction in robotic manipulation and focus on the routing problem assuming that the user-defined costs abstract away the operational difficulties of obtaining additional views when a multi-view reconstruction model is chosen. The appropriate choice between Image-to-3D and view-invariant reconstruction therefore depends on this deployment context: a pipeline running on multi-view-friendly hardware may default to view-invariant reconstruction, a pipeline with limited view-acquisition capability may rely primarily on Image-to-3D models, and many systems benefit from per-input selection across both regimes.

2.3 Model Routing for Large Language Models

Exhaustively querying every reconstruction model for each input is computationally intractable, inspiring a routing function that selects a single model per input. To the best of our knowledge, routing has not been explored for 3D reconstruction; however, it has emerged as an active research area for Large Language Models (LLMs), where LLMs with complementary strengths motivate per-query selection. Early routing approaches focused on learning fixed mappings from queries to models. **ZOOTER** [26] trains a lightweight router through knowledge distillation from a reward model. To mitigate noise in per-query reward estimates, ZOOTER introduces a tag-based label enhancement procedure that smooths each per-query score towards the average score within its category (*e.g.* math problems, creative writing), under the assumption that model rankings are influenced by both the stable prompt category and noisy individual query features. **RouterDC** [27] learns per-model embeddings using dual contrastive losses: a sample-LLM loss that aligns queries with high-performing models while repelling them from poor-performing ones, and a sample-sample loss that clusters similar queries with the purpose of improving training stability. Both methods achieve strong routing performance, yet they train under a fixed cost function, preventing a user from weighting cost versus accuracy differently at inference time without retraining. Subsequent work addressed this limitation through the development of routers capable of operating across arbitrary cost functions at inference time. **RouterBench** [28] introduces standardized baselines and evaluation metrics — including Average Improvement in Quality (AIQ) and deferral curves — for routers that are able to adjust their behavior based on user-specified cost constraints. **RouteLLM** [29] learns binary routers using matrix factorization over query embeddings and learnable per-model embeddings, supporting different cost-performance preferences via a tunable threshold. Li [30] systematically compares non-parametric and parametric routing approaches, finding that a well-tuned k-Nearest Neighbors approach matches or outperforms more complex learned methods, attributing the improved performance to locality properties in embedding spaces. These findings inform our choice of baselines in Chapter 5. While the

LLM routing literature provides useful conceptual scaffolding, it does not directly transfer to 3D reconstruction due to important structural differences between the two. We discuss these distinctions formally in Chapter 3.

2.4 Reconstruction-Based Manipulation Pipelines

A large number of manipulation pipelines treat 3D reconstruction as a core perception module, using the reconstructed mesh to plan grasps, simulate contact, or condition downstream policies. **SceneComplete** [17] reconstructs complete, segmented 3D models of cluttered scenes from a single RGB-D view, producing meshes that support downstream grasp planning. **ZeroGrasp** [19] jointly predicts 3D reconstructions and 6-DoF grasp poses from a single RGB-D image, using the reconstructed shape to refine grasp poses. **RoboTwin** [42] generates simulation-ready object meshes from single 2D images via 3D generative foundation models, which are then used to pre-train dual-arm manipulation policies. **ManiTwin** [43] produces 3D assets annotated for manipulation from single images, supporting large-scale policy learning across diverse object categories. **SuperQ-GRASP** [40] builds object meshes from a multi-view RGB scan via a NeRF-based reconstruction step and decomposes the mesh into superquadric primitives, which are then used for grasp pose estimation. **YOSO** [41] introduces a pipeline in which a robotic arm scans the workspace from multiple viewpoints, reconstructs per-object meshes from the captured RGB-D images, and merges these meshes with the initial partial point cloud of the scene to ultimately predict grasps. Each of these systems commits to a single reconstruction approach upfront. Further, recent empirical evaluation by Nolte et al. [13] examines whether current single-view reconstruction methods are suitable for robotic manipulation, concluding that current Image-to-3D models fail to meet robotics-specific requirements in their evaluation.

Choosing a single reconstruction approach exposes the downstream pipeline to two related vulnerabilities. The first is a *robustness* concern: when the chosen reconstruction approach fails on a particular input—producing an inaccurate mesh due to viewpoint, texture, or category bias—the downstream task inherits that failure. The second is an *efficiency* concern: Image-to-3D models impose latency and memory costs that may exceed the budget of a real-time deployment, and multi-view methods impose view-acquisition costs. Per-input model routing addresses both concerns simultaneously: by selecting models per-input under user-specified cost constraints, the manipulation pipeline can avoid systematic exposure to any single model’s failure modes while allocating computational and view-acquisition resources only when input difficulty warrants it. To the best of our knowledge, this combination of input-aware routing for robustness and efficiency in reconstruction-based manipulation has not previously been pursued. The remainder of this thesis develops a framework designed to realize this capability.

Chapter 3

Problem Formulation

This chapter formalizes the model routing problem for 3D reconstruction. We define the routing objective, specify the structure of scores and costs, and outline how a router is evaluated. We then contrast our model routing setting for reconstruction methods with a setting for routing LLMs, which differs in important structural ways.

3.1 Setup

Let $\mathcal{M} = \{m_1, \dots, m_k\}$ denote a set of k candidate 3D reconstruction models, each producing a mesh from an input image $x \in \mathcal{X}$ taken from the current robot viewpoint. Given x , we seek to select a model $\hat{m} \in \mathcal{M}$ whose reconstruction is most useful for the downstream task under user-specified resource constraints. The routing problem is characterized by two functions:

- A **score function** $s : \mathcal{X} \times \mathcal{M} \rightarrow \mathbb{R}$ that measures the quality of model m 's reconstruction on input x , where higher values indicate better quality. The score function is metric-specific: for any choice of mesh-quality metric or intrinsic mesh property (*e.g.*, IoU, Chamfer distance, DCD, or Eval3D structural soundness), s is the negated or normalized form of that metric so that higher values consistently indicate better quality. The specific metrics evaluated in this thesis are described in Section 5.1.3.
- An **image-independent cost function** $c : \mathcal{M} \rightarrow \mathbb{R}_{\geq 0}$ that captures a user-specified penalty for each model. The cost function is expressed in the same units as the score, so that the utility $u = s - c$ is dimensionally consistent. In practice, the user specifies c by translating raw operational costs (*e.g.*, latency, memory, energy) into a single per-model penalty that reflects their deployment priorities.

A router h is a function that takes an input image x and a cost function c , and outputs a selected model $\hat{m} = h(x, c)$. The router's goal is to maximize the *utility* of its selection, defined as the cost-adjusted score of the chosen model:

$$u(x, \hat{m}; c) = s(x, \hat{m}) - c(\hat{m}).$$

We assume access to a training dataset $\mathcal{D}_{\text{train}} = \{(x^{(i)}, \mathbf{s}^{(i)})\}_{i=1}^n$ of input-score pairs, where $\mathbf{s}^{(i)} \in \mathbb{R}^k$ denotes the vector of reconstruction scores $s(x^{(i)}, m_j)$ for $j \in [k]$. The router h is learned from $\mathcal{D}_{\text{train}}$ and evaluated on a held-out set of inputs at inference time.

3.2 Score and Cost functions

Score function. The score function $s(x, m)$ measures reconstruction quality with respect to a single chosen mesh-quality metric. In practice, the score is either computed against a ground-truth mesh using one of several mesh-quality metrics, such as Chamfer distance variants or IoU, or is computed solely from properties of the generated mesh using metrics from Eval3D [35]. Different metrics lead to different optimal routing decisions: IoU is volume-based and strongly penalizes hollow geometry, whereas Chamfer-based metrics measure surface correspondences, and different downstream manipulation tasks may favor distinct quality criteria. For instance, a renderer may prefer structurally sound meshes as measured by Eval3D-struct, while a contact-rich dexterous task may prefer fine surface detail as measured by Density-aware Chamfer Distance (DCD) [34]. We therefore do not commit the framework to a single metric. A separate routing model is trained for each metric, and the user selects the metric at training time based on the downstream task. The framework’s contribution is learning an effective router for any chosen metric, not learning to predict multiple metrics jointly.

Cost function. The cost function $c(m)$ encodes user preferences over per-model costs. Examples of relevant cost dimensions include inference latency, peak memory usage, monetary expense (for paid APIs), and energy consumption. Robotics deployments often face multiple simultaneous resource constraints — a robot operating on edge hardware may need to penalize both latency and memory, while a cloud-deployed pipeline may primarily care about monetary cost.

The cost function c is specified at inference time and is not needed during training, ensuring that the routing framework supports arbitrary user-specified cost penalizations without requiring retraining. This allows a single trained router to serve a robot prioritizing latency, a robot prioritizing memory, or any weighted combination of the two. Such flexibility is essential for deployment scenarios where cost preferences vary across robots, tasks, or even temporally within a single deployment (*e.g.*, switching between high-throughput and low-power modes).

Beyond representing soft preferences, costs can also encode hard feasibility constraints. Setting $c(m_j) = \infty$ effectively excludes model m_j from consideration, which is useful when a model’s resource demands exceed the robot’s available budget (*e.g.*, a model that requires more GPU memory than the robot has) or when kinematic limitations make view-invariant methods infeasible. The router’s behavior under such constraints should degrade gracefully: when a preferred model is infeasible, the router should select the next-best feasible alternative rather than fail.

3.3 Routing objective

The router’s selection on input x under the cost function c is $h(x, c)$. We define the *oracle* as the router that has access to the ground-truth scores $\mathbf{s}^{(i)}$ and selects

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

The oracle represents an upper bound on what any router can achieve. Following standard practice in utility evaluation [28,30], we evaluate a router by its expected *regret* relative to the oracle on a held-out set of n examples:

$$\mathcal{R}(h; c) = \frac{1}{n} \sum_{i=1}^n \left[\max_{m \in \mathcal{M}} (s(x^{(i)}, m) - c(m)) - (s(x^{(i)}, h(x^{(i)}, c)) - c(h(x^{(i)}, c))) \right]. \quad (3.1)$$

Lower regret indicates a better router. Reporting regret rather than absolute utility serves two purposes. First, it normalizes performance to the oracle, allowing comparison across different cost functions, datasets, and quality metrics. Second, it weighs routing mistakes by their severity. To illustrate, choosing a slightly suboptimal model when two models perform nearly identically incurs small regret, whereas choosing a substantially worse model incurs large regret. This makes regret a more informative signal than selection accuracy, which would treat both cases as equivalent errors.

3.4 Comparison with LLM Routing

Model routing has been studied extensively for Large Language Models (LLMs), with several methods proposed to select among the many frontier LLMs based on input properties [26–30]. Prior work [30] formalizes the most general LLM routing problem as maximizing the following objective for a user-defined scalar λ and specific cost function c_{API} that represents API costs:

$$m^* = \arg \max_{m \in \mathcal{M}} s(x, m) - \lambda \cdot c_{\text{API}}(x, m).$$

Our formulation differs in three ways: the dependence of c on just m rather than on both x and m , the need to be robust to arbitrary cost functions c rather than just the single API-pricing-based cost function, and the composition of the model pool $\mathcal{M}$. We discuss each in turn.

Input-independent costs. In the LLM setting, inference cost is a function of the number of input and output tokens [44], both of which depend on the input prompt. The cost function therefore depends on both the input prompt and the queried model. In contrast, 3D reconstruction models typically employ architectures with fixed computational graphs (*e.g.*, pure feed-forward [7] or fixed-step diffusion pipelines [9]), resulting in a constant inference cost regardless of the input. 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 as well [28,30]. This simplification allows the cost function to be specified without requiring online per-input cost estimation.

Multi-dimensional cost structures. The LLM routing problem originated from selecting between paid API endpoints, where costs are naturally expressed in dollars per API call. Routing between 3D reconstruction models in robotics requires a more task-specific and often multi-dimensional cost structure. A robot may simultaneously care about inference latency and peak memory usage, and depending on the constraints of the robot task, one of these factors may carry more weight in some runs and less weight in others. Further, the cost structure can be considerably more complex in practice: in some applications, a robot may assign a cost of ∞ to models exceeding its memory budget, resulting in a binary

feasibility constraint. We therefore argue that a routing framework for 3D reconstruction must be robust to any cost function c , whereas a routing framework for LLMs only needs to be robust to the cost function representing API pricing.

Heterogeneous model pool. LLM routing considers only generative language models, which form a relatively homogeneous pool: each model takes a text prompt and produces a text response. The 3D reconstruction model pool is structurally more diverse. *Viewpoint-dependent* models, such as Image-to-3D models, produce scores that depend on both the model itself and the input image representing the initial robot view as a frontal view can yield far worse reconstructions than a corner view of the same object. *View-invariant* methods aggregate information across multiple captured viewpoints, so their reconstruction quality depends on object-level properties rather than on the robot’s initial view. This heterogeneity in the model pool motivates SCOUT’s architectural design, which we describe in Chapter 4.

Chapter 4

SCOUT

This chapter develops SCOUT (**S**core-**C**onditioned **O**ptimal **U**tility **T**argeting), our routing framework for 3D reconstruction. SCOUT is designed for two properties: (1) *scalability*: the learned network’s (Section 4.2) training and performance remain independent of the view-invariant pool; and (2) *flexibility*: routing under arbitrary cost functions. The framework consists of three stages: preprocessing, score decoupling, and utility optimization.

4.1 Preprocessing

Let $\mathbf{s}^{(i)} = (s_1^{(i)}, \dots, s_k^{(i)})$ 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

$$\mathbf{s}^{(i)} \equiv \begin{bmatrix} \mathbf{s}_{\text{dep}}^{(i)} \\ \mathbf{s}_{\text{inv}}^{(i)} \end{bmatrix}$$

with $\mathbf{s}_{\text{dep}}^{(i)} \in \mathbb{R}^{k_1}$, $\mathbf{s}_{\text{inv}}^{(i)} \in \mathbb{R}^{k_2}$, and $k_1 + k_2 = k$. This partition reflects the structural distinction discussed in Chapter 3: viewpoint-dependent scores depend on the input image, while view-invariant scores depend on object-level properties rather than the robot’s initial view.

Following ZOOTOER [26] and motivated by the variance in reconstruction quality across different object categories, we apply tag-based score smoothing to $\mathbf{s}_{\text{dep}}^{(i)}$. We group objects into semantic categories using a tagger $\mathcal{T} : \mathcal{X} \rightarrow \text{tag}$ 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 tag set is constructed once per dataset by hierarchically prompting a vision-language model to propose a small set of semantically meaningful object categories from the training images, after which the same model assigns each input to one of these categories (or to “other” if none apply). The smoothed scores are

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

where $\beta \in [0, 1]$ is a hyperparameter that controls the degree of smoothing. Smoothing is applied only when learning relative model performance, as described in Section 4.2. The final recovered scores aim to estimate the original unsmoothed values $\mathbf{s}_{\text{dep}}^{(i)}$, which are needed for comparison with $\mathbf{s}_{\text{inv}}^{(i)}$.

4.2 Score Decoupling

Given the smoothed scores $\tilde{s}_{\text{dep}}^{(i)}$, we now define the main technical contribution of SCOUT: decoupling the predicted scores into two interpretable components: (1) the relative performance of the viewpoint-dependent models, represented by a learned probability distribution $\hat{\mathbf{p}}$, and (2) the difficulty of the input image, estimated by a scalar partition function $\hat{z}$. This decoupling preserves the relative utilities among viewpoint-dependent models while enabling direct comparison with view-invariant scores $\mathbf{s}_{\text{inv}}$.

The smoothed scores $\tilde{s}_{\text{dep}}^{(i)}$ can be predicted using several possible training objectives, including MSE directly with the smoothed scores or KL divergence on the softmax distribution of the smoothed scores. We use the KL divergence formulation given that an easy input image (*e.g.*, a corner view) tends to yield high scores across all viewpoint-dependent models, while a hard image (*e.g.*, a frontal view) tends to yield 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 viewpoint-dependent models by normalizing out the 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 as seen in Figure 4.1.

For each input image, we convert the smoothed scores into a probability distribution via the softmax with temperature T :

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

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

Recovering the unsmoothed scores from $\hat{\mathbf{p}}$ requires estimation of a partition function. We define

$$z^{(i)} = \sum_{m=1}^{k_1} e^{s_{\text{dep},m}^{(i)}/T}$$

over the original unsmoothed scores, giving us $T \cdot \ln(\hat{p}_j^{(i)} \cdot z^{(i)})$ as an approximation for $s_{\text{dep},j}^{(i)}$. If only viewpoint-dependent models were present, $T \cdot \ln(\hat{\mathbf{p}})$ could be used directly for utility optimization, as all scores across each model for input $x^{(i)}$ would be offset by the same constant $T \cdot \ln(z^{(i)})$. However, given that viewpoint-dependent scores must be compared against the view-invariant scores $\mathbf{s}_{\text{inv}}$ as well, an estimate of $\hat{z}$ (as defined above) is required to recover absolute score values.

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

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

rather than using $z^{(i)}$ directly. Although $\hat{\mathbf{p}}$ is trained on the softmax of smoothed scores, the recovered scores $\hat{s}(x^{(i)}, m_j) = T \cdot \ln(\hat{p}_j^{(i)} \cdot \hat{z}^{(i)})$ are intended to approximate the true unsmoothed

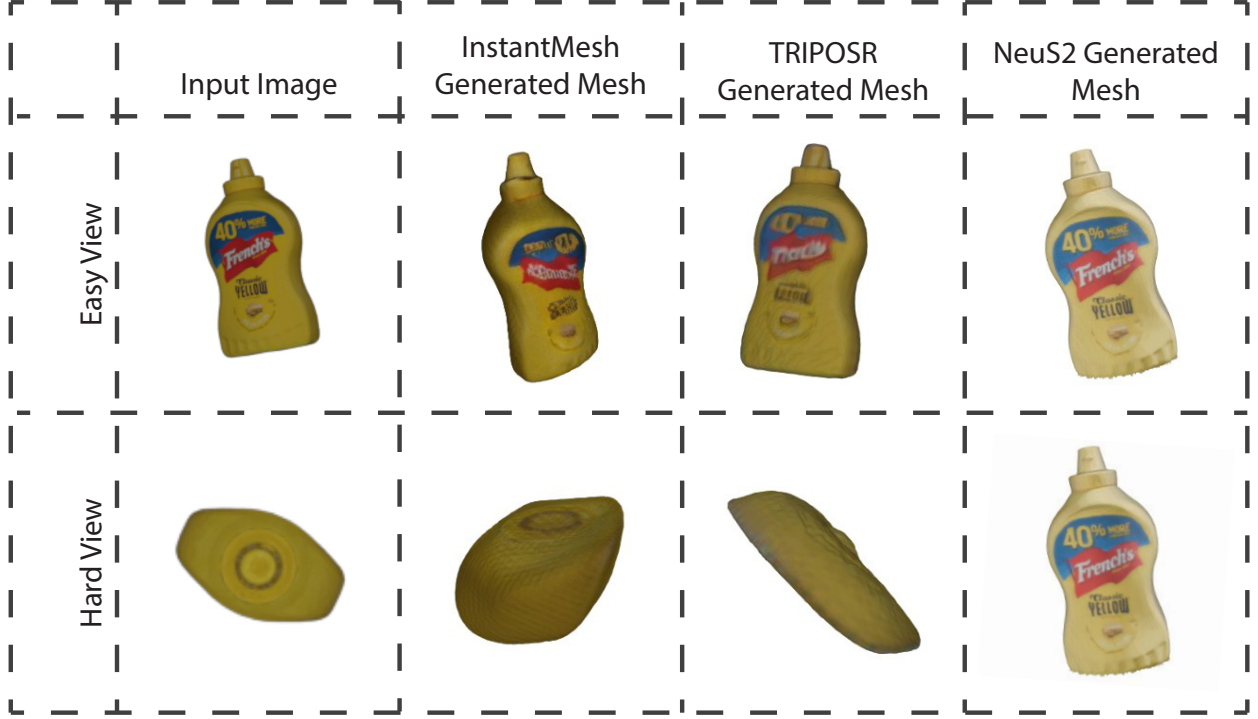


Figure 4.1: InstantMesh and TripoSR (viewpoint-dependent) produce high-quality reconstructions on the easy input (top) and degraded reconstructions on the hard input (bottom)—their scores share a common difficulty-induced offset. NeuS2 (view-invariant) captures its own views during reconstruction, so the quality of its produced mesh is uncorrelated with the input image’s difficulty. A single softmax over all three models would conflate these two regimes; decoupling separates them.

scores $s_{\text{dep},j}^{(i)}$. The proxy construction above ensures this targeting by design as

$$T \cdot \ln(\hat{p}_j^{(i)} \cdot \tilde{z}_j^{(i)}) = T \cdot \ln\left(\hat{p}_j^{(i)} \cdot \frac{e^{s_{\text{dep},j}^{(i)}/T}}{\hat{p}_j^{(i)}}\right) = s_{\text{dep},j}^{(i)}.$$

The intent of smoothing is to minimize noise in the training targets for $\hat{\mathbf{p}}$ in order to produce more accurate predictions of relative model performance. However, the smoothed scores $\tilde{s}_{\text{dep}}^{(i)}$ themselves are not the quantities that we aim to actually recover. Smoothing is *not* applied during the construction of the proxy targets for $\hat{z}$ as $\hat{z}$ must capture the absolute difficulty of each individual input image, and smoothing suppresses this per-image variation through the blending of scores towards the category averages. This setup ensures that each proxy recovers exactly the true score for model j , regardless of what $\hat{p}_j^{(i)}$ was trained on, as the proxy is derived directly from the true scores. This yields k_1 proxies for $z^{(i)}$, which will then be combined into a single estimate, since all viewpoint-dependent scores are offset by the same constant $T \cdot \ln(z)$.

The optimal combination for a target representation of z is a weighted average:

$$\tilde{z}^{(i)} = \sum_{j=1}^{k_1} w_j \tilde{z}_j^{(i)}, \quad w_j \propto \frac{1}{S_j^2(1 - R_j^2) \mathbb{E}_i \left[\frac{z^4}{e^{2s_{\text{dep},j}/T}} \right]} \quad (4.1)$$

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 complete derivation, based on minimizing the variance of the combined estimator under a noise model on $\hat{\mathbf{p}}$, is presented in Section 4.4 at the end of this chapter. The resulting $\tilde{z}^{(i)}$ values serve as regression targets for the second supervised model that produces $\hat{z}$.

The derived equation weights model j according to two main features. It downweights model j when (i) its predicted probabilities $\hat{p}_j$ are noisy (large $S_j^2(1 - R_j^2)$), or (ii) its true probability is frequently low, making the proxy $\tilde{z}_j^{(i)}$ a higher-variance estimate of $z^{(i)}$. We compare this weighting against three natural 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 find that the derived weights improve upon all alternatives across the majority of evaluation metrics (Section 5.4.2).

4.3 Utility Optimization

Given the predicted probability distribution $\hat{\mathbf{p}}$ and the estimated partition function $\hat{z}$, the optimal reconstruction model is defined as $\hat{m} = \arg \max_{m_j \in \mathcal{M}} \hat{s}(x, m_j) - c(m_j)$, where the estimated score is defined as

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

To learn $\hat{s}_{\text{inv}}(m_j)$, we use simple input features such as the mean score within each input's tag-based category, capturing object-level priors, or linear regression on a feature embedding of the input image, capturing image-level properties such as texture and reflectance (further elaborated in Section 5.1.4).

Further, in contrast to the decoupling, 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 Figure 4.2.

4.4 Derivation of Optimal Weights

This section derives the optimal weights used in Equation (4.1). For brevity, we refer to $s_{\text{dep},j}$ as just s_j for the remainder of this section. The goal is to derive global inverse-variance weights for combining k_1 estimators of the partition function. Each method $j \in \{1, \dots, k_1\}$

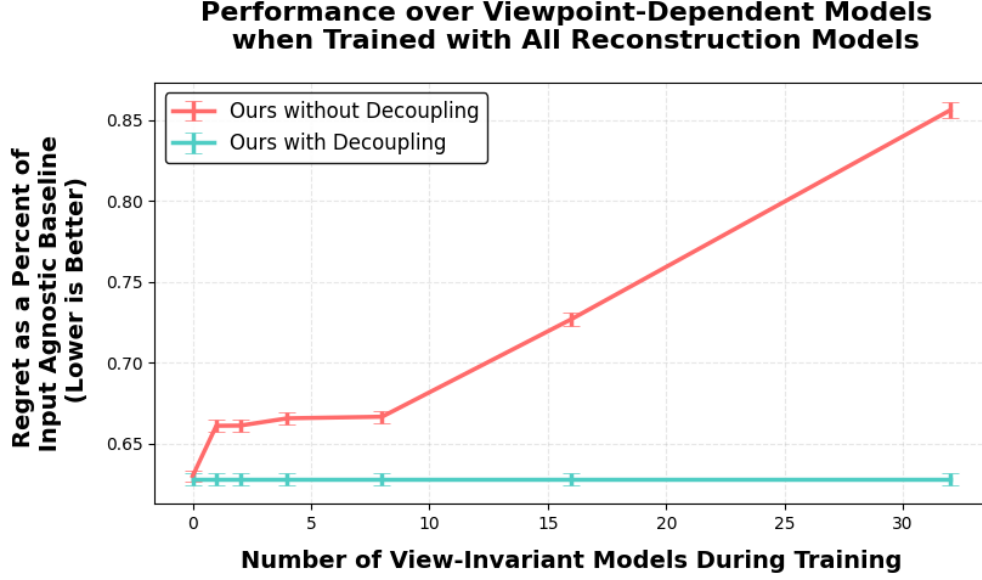


Figure 4.2: Effect of the number of view-invariant methods on routing performance when evaluated on viewpoint-dependent methods. With decoupling, regret over viewpoint-dependent models remains constant regardless of how many view-invariant methods are included during training. Without decoupling, regret increases as the number of view-invariant methods grows.

produces, for data point i , the estimate

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{\hat{p}_j^{(i)}}, \quad (4.2)$$

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

$$\tilde{z}^{(i)} = \sum_{j=1}^{k_1} w_j \tilde{z}_j^{(i)}. \quad (4.3)$$

4.4.1 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 $\text{Var}_\epsilon[\epsilon_j^{(i)}] = \sigma_j^2$. 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:** $\text{Cov}_\epsilon[\epsilon_j^{(i)}, \epsilon_k^{(i)}] = 0$ 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 .

4.4.2 Global weight optimization

Fixing a data point i and expanding Equation 4.2 to first order in $\epsilon_j^{(i)}/p_j^{*(i)}$:

$$\tilde{z}_j^{(i)} = \frac{e^{s_j^{(i)}/T}}{p_j^{*(i)} + \epsilon_j^{(i)}} \approx z^{(i)} - \frac{z^{(i)}}{p_j^{*(i)}} \epsilon_j^{(i)}. \quad (4.4)$$

Hence

$$\text{Var}_\epsilon[\tilde{z}_j^{(i)}] = \left(\frac{z^{(i)}}{p_j^{*(i)}} \right)^2 \sigma_j^2. \quad (4.5)$$

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

$$\tau_j^{2,(i)} \equiv \text{Var}_\epsilon[\tilde{z}_j^{(i)}] = \frac{(z^{(i)})^4}{e^{2s_j^{(i)}/T}} \cdot \sigma_j^2. \quad (4.6)$$

Inserting Equation 4.4 into Equation 4.3:

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

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

$$\text{Var}_\epsilon[\tilde{z}^{(i)}] = \sum_j w_j^2 \tau_j^{2,(i)}. \quad (4.7)$$

The quantity in Equation 4.7 depends on i through $\tau_j^{2,(i)}$. Since we seek weights independent of i , we minimize the variance *averaged* over data points:

$$\begin{aligned} J(\mathbf{w}) &= \mathbb{E}_i[\text{Var}_\epsilon[\tilde{z}^{(i)}]] \\ &= \mathbb{E}_i\left[\sum_j w_j^2 \tau_j^{2,(i)}\right] \\ &= \sum_j w_j^2 \mathbb{E}_i[\tau_j^{2,(i)}] \\ &\equiv \sum_j w_j^2 \bar{\tau}_j^2. \end{aligned} \quad (4.8)$$

Substituting Equation 4.6:

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

We minimize Equation 4.8 subject to $\sum_j w_j = 1$. The Lagrangian is

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

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

$$\sum_j \frac{\lambda}{2\bar{\tau}_j^2} = 1 \quad \implies \quad \lambda = \frac{2}{\sum_k 1/\bar{\tau}_k^2}.$$

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

$$w_j = \frac{1/\bar{\tau}_j^2}{\sum_k 1/\bar{\tau}_k^2} \propto \frac{1}{\bar{\tau}_j^2} = \frac{1}{\sigma_j^2 \mathbb{E}_i \left[\frac{(z^{(i)})^4}{e^{2s_j^{(i)}/T}} \right]}. \quad (4.9)$$

4.4.3 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:

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

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

$$S_j^2 = \text{Var}_i[p_j^{*(i)} + \epsilon_j^{(i)}] = \text{Var}_i[p_j^{*(i)}] + \sigma_j^2. \quad (4.10)$$

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

$$R_j^2 = \frac{\text{Cov}_i[p_j^*, \hat{p}_j]^2}{\text{Var}_i[p_j^*] \cdot S_j^2} = \frac{\text{Var}_i[p_j^*]^2}{\text{Var}_i[p_j^*] \cdot S_j^2} = \frac{\text{Var}_i[p_j^*]}{S_j^2}.$$

Substituting $\text{Var}_i[p_j^{*(i)}] = R_j^2 S_j^2$ into Equation 4.10:

$$\sigma_j^2 = S_j^2(1 - R_j^2). \quad (4.11)$$

Substituting Equation 4.11 back into Equation 4.9 gives the form used in Equation 4.1, completing the derivation.

Chapter 5

Experiments

We evaluate SCOUT across two axes: routing performance against adapted baselines on benchmark datasets (Section 5.2), and downstream impact on robotic manipulation tasks (Section 5.3). We additionally validate the design choices in SCOUT through ablations (Section 5.4) and analyze what the learned router has captured (Section 5.5).

5.1 Experimental Setup

5.1.1 Mesh Reconstruction Models

We evaluate SCOUT with eight reconstruction models. Four are viewpoint-dependent Image-to-3D models selected for architectural diversity: TripoSR [7], InstantMesh [9], Hunyuan3D 2.0 [12], and TRELIS [11]. Three are view-invariant methods covering Gaussian (2DGS [23]), NeRF (NeRF2Mesh [21]), and SDF (NeuS2 [22]) representations. The eighth option is *skip*, which bypasses reconstruction with a fixed default score for time-sensitive settings. *Skip* is also considered a view-invariant method as the reconstruction score is independent of the initial robot view. We refer to Sections 2.1 and 2.2 for detailed descriptions of each model.

5.1.2 Datasets

We evaluate on two datasets that capture complementary regimes of single-image reconstruction difficulty.

Google Scanned Objects (GSO) [31] provides ground-truth meshes for 1,030 everyday household objects scanned at high resolution. We render 15,450 images by sampling each object at multiple viewpoints. The viewpoints of the GSO dataset are chosen to be nondegenerate by avoiding face-on views that show only a single planar surface. The non-degeneracy of the images make the image-to-3D problem easier than if we were to capture face-on views.

YCB [33] provides ground-truth meshes for manipulation-relevant objects (cans, cups, tools, food packaging). We chose 54 of these objects that have corresponding real photographs from the BigBIRD dataset [32]. We capture each object across multiple viewpoints—often deliberately face-on—and across three image styles for each viewpoint: real BigBIRD photographs, flash-lighting renders, and surround-lighting renders. This yields 2,106 images that probe reconstruction under the harder conditions a robot is likely to face: degenerate

viewpoints, real-world image artifacts, and varied lighting. We refer to this combined dataset as BigBIRD + YCB throughout.

5.1.3 Mesh Quality Metrics

The score function $s(x, m)$ in our problem formulation is implemented as a mesh quality metric that is either computed against the ground-truth mesh or is a property of the reconstructed mesh itself. Each reconstructed mesh is first registered and scaled into a canonical frame (described in Section 5.1.8), then scored using one of seven metrics:

- **Chamfer L1** and **Chamfer L2**: bidirectional point-to-point distance between mesh surfaces, under L_1 and L_2 norms.
- **Density-aware Chamfer Distance (DCD)** [34]: a Chamfer variant that downweights regions where the predicted point density mismatches the ground truth, addressing Chamfer’s sensitivity to outliers.
- **Intersection over Union (IoU)**: volumetric overlap between the predicted and ground-truth meshes.
- **Earth Movers Distance-Maximum Mean Discrepancy (MMD-EMD)**: an approximate optimal transport distance between point sets sampled from the predicted and ground-truth meshes [45,46].
- **Eval3D-geo** and **Eval3D-struct** [35]: geometric and structural metrics from the Eval3D benchmark, which capture the plausibility of the reconstructed mesh.

For all metrics, we negate the values where needed so that higher is better, conforming to the convention $s(x, m) \in \mathbb{R}$, with higher values indicating better reconstruction.

5.1.4 Predicting Scores for View-Invariant Methods

The view-invariant scores s_j for $m_j \in \mathcal{M}_{\text{inv}}$ enter the SCOUT optimization (Section 4.3) as inputs and must be predicted for comparison with the scores estimated for the viewpoint-dependent reconstruction methods. We considered two candidate predictors and selected between them per metric based on validation performance:

- **Linear regression on image embeddings.** A ridge regression model maps the image embedding directly to each view-invariant model’s score. This predictor is well-suited to metrics where score variation reflects learnable texture or geometric properties of the input.
- **Category-based means.** For each object category in the training set, we compute the mean score of each view-invariant model and use this as the predicted score for novel objects in that category. This predictor exploits object priors and assumes that within-category score variation is small or random.

Linear regression was selected for GSO DCD as well as for BigBIRD + YCB Chamfer L2 and MMD-EMD. Category-based means were selected for the remaining five BigBIRD + YCB metrics. The selection pattern reflects a structural property of view-invariant reconstruction: because these methods capture multiple views of the object during reconstruction (Section 2.2), their output quality is largely independent of any single input view, resulting in the score distribution often having less learnable variation than the that of the score distribution of the viewpoint-dependent methods.

Within an object category, view-invariant scores still exhibit variance across individual objects. This within-category variance arises from two distinct sources. The first is random mesh-optimization artifacts — small surface noise, occasional spurious holes — which are often unlearnable from the input image. The second is systematic image-conditional signal arising from object properties that affect reconstruction, such as surface color, reflectance, or texture. The relative contribution of these two sources varies by metric and dataset, and the appropriate predictor follows accordingly. When within-category variance is dominated by artifacts, linear regression on image features adds noise rather than signal, and the category-based mean is the better unbiased predictor. When within-category variance contains exploitable image-conditional signal—for example, the GSO dataset contains both untextured black shoes and multi-colored bright shoes, whose material differences systematically affect view-invariant reconstruction quality—linear regression on image embeddings can recover this signal and outperform the category-based mean.

Table 5.1 quantifies this property. For each metric, we compute the within-category mean absolute deviation (MAD) of each model’s scores, then average across models within each viewpoint-handling group. Lower MAD indicates that scores are more concentrated within an object category, which is the regime where category-based means succeed. Across the seven BigBIRD + YCB metrics, view-invariant scores have lower within-category MAD on five (DCD, Chamfer, L1, IoU, Eval3D-geo, Eval3D-struct). The two BigBIRD + YCB exceptions (Chamfer, L2, MMD-EMD), together with GSO DCD, coincide exactly with the three metrics for which we selected linear regression. The MAD gap is consistent but modest because unlearnable random noise (such as randomly produced internal artifacts) places a floor on how concentrated scores can be within a category.

5.1.5 Cost Families for Evaluations

A central design property of SCOUT is support for arbitrary cost functions c at inference time. Let $\mathcal{M}_{\text{dep}} \subseteq \mathcal{M}$ denote the set of viewpoint-dependent models and $\mathcal{M}_{\text{inv}} \subseteq \mathcal{M}$ the set of view-invariant methods, so that $\mathcal{M} = \mathcal{M}_{\text{dep}} \cup \mathcal{M}_{\text{inv}}$. We evaluate over five cost regimes that capture distinct deployment scenarios.

Viewpoint-dependent-only zero-cost function:

$$c_{\text{view, no-cost}}(m) = \begin{cases} 0 & m \in \mathcal{M}_{\text{dep}} \\ \infty & m \in \mathcal{M}_{\text{inv}} \end{cases}$$

Viewpoint-dependent models incur no cost while view-invariant methods are excluded entirely. This regime enables direct comparison with ZOOTHER and RouterDC, which do not support varying costs at inference time.

Dataset	Metric	View-dep. MAD	View-inv. MAD	Ratio (dep/inv)
GSO	DCD	0.0622	0.0884	0.70
BigBIRD + YCB	DCD	0.0850	0.0508	1.67
BigBIRD + YCB	Chamfer L2	0.0461	0.0827	0.56
BigBIRD + YCB	Chamfer L1	0.0526	0.0411	1.28
BigBIRD + YCB	IoU	0.1837	0.1254	1.47
BigBIRD + YCB	MMD-EMD	0.0255	0.0297	0.86
BigBIRD + YCB	Eval3D-geo	0.0527	0.0376	1.40
BigBIRD + YCB	Eval3D-struct	0.0823	0.0656	1.25

Table 5.1: Within-category mean absolute deviation (MAD) of scores for the two viewpoint-handling groups. For each metric, we group (model, view) pairs by object category, compute the MAD around the per-category mean within each category, weight-average across categories by category size, and average across models in each group. A ratio > 1 (bold) indicates that view-dependent scores are more spread within category than view-invariant scores.

Zero cost function:

$$c_{\text{no-cost}}(m) = 0, \quad \forall m \in \mathcal{M}$$

The router selects purely on quality, with no penalization of any model. This regime tests pure routing accuracy, independent of cost-quality tradeoffs. This regime also enables direct comparison with ZOOPER and RouterDC.

Full cost-function family $\mathcal{C}$: a family of cost functions where each model m_j is assigned a cost drawn from a model-specific interval. Define

$$[a_j, b_j] = \left[P_{75,j} - \min_{j'}(\text{IQR}_{j'}), P_{75,j} \right],$$

where $P_{75,j}$ and IQR_j are the 75th percentile and interquartile range of model m_j 's scores. The family $\mathcal{C}$ is then the set of all cost functions $c : \mathcal{M} \rightarrow \mathbb{R}_{\geq 0}$ such that

$$c(m_j) \in [a_j, b_j] \quad \text{for every } m_j \in \mathcal{M}.$$

The intervals $[a_j, b_j]$ are chosen to keep all models in comparable utility ranges, ensuring that no single model is strictly dominant under typical cost functions and forcing the router to make non-trivial routing decisions. We exclude the *skip* model when computing $\min_{j'}(\text{IQR}_{j'})$, since its fixed score yields a zero interquartile range.

Viewpoint-dependent-only cost-function family $\mathcal{C}_{\text{dep}}$: the analogous family in which view-invariant models are excluded. Specifically,

$$\mathcal{C}_{\text{dep}} = \{c : \mathcal{M} \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\} \mid c(m_j) \in [a_j, b_j] \text{ for } m_j \in \mathcal{M}_{\text{dep}}, c(m_j) = \infty \text{ for } m_j \in \mathcal{M}_{\text{inv}}\}.$$

This combines the multi-dimensional structure of $\mathcal{C}$ with the exclusion of view-invariant methods.

One-dimensional cost families: for comparison with the LLM routing setting, we additionally evaluate over three families of cost functions, each defined by scaling a fixed cost

function by a scalar $\lambda \geq 0$. Specifically, given a fixed cost function $c_{\text{fixed}} : \mathcal{M} \rightarrow \mathbb{R}_{\geq 0}$, the corresponding one-dimensional family is

$$\mathcal{C}_{c_{\text{fixed}}} = \{\lambda \cdot c_{\text{fixed}} \mid \lambda \in \mathbb{R}_{\geq 0}\}.$$

The scalar λ parametrizes the cost-quality tradeoff: $\lambda = 0$ recovers the zero cost function (pure quality), while increasing λ progressively penalizes expensive models. We evaluate three such families, defined by three choices of c_{fixed} : latency-only ($c_{\text{latency}}(m) = \text{latency}(m)$), memory-only ($c_{\text{memory}}(m) = \text{memory}(m)$), and the per-model product of latency and memory, corresponding to memory-time occupancy in GB-s ($c_{\text{latency} \times \text{memory}}(m) = \text{latency}(m) \cdot \text{memory}(m)$). The specific latency and memory values used to construct these families are given in Section 5.1.8.

5.1.6 Baselines

To the best of our knowledge, no prior routing methods exist for 3D reconstruction. We therefore implement baselines based on LLM routing approaches. All baselines and SCOUT share the same pretrained image features (described in Section 5.1.8) for fair comparison.

Input-agnostic baseline. Selects $\arg \max_{m_j} \bar{s}(m_j) - c(m_j)$ using each model’s training-set mean score $\bar{s}(m_j)$, ignoring the input image entirely. This baseline isolates whether per-input selection helps over a fixed model preference. We report all routing results relative to the input-agnostic baseline’s regret, allowing comparison across cost function regimes and quality metrics.

ZOOTER [26]. Trains a router via knowledge distillation from a reward model with tag-based label enhancement, as discussed in Section 2.3. ZOOTER does not natively support varying the cost function at inference time, so it can only be evaluated on the zero-cost regimes $c_{\text{no-cost}}$ and $c_{\text{view, no-cost}}$.

RouterDC [27]. Learns per-model embeddings using dual contrastive losses. Like ZOOTER, RouterDC does not support varying the cost function at inference and is restricted to the zero-cost regimes.

MLP. A multi-layer perceptron baseline from RouterBench [28] that predicts per-model utilities directly from image features. This baseline supports arbitrary cost functions at inference.

Matrix factorization (MF). A matrix factorization baseline based on RouteLLM [29] that learns low-rank query-model interactions. This baseline supports arbitrary cost functions at inference.

k-Nearest Neighbors (kNN). The non-parametric baseline shown by Li [30] to outperform more complex learned routers in LLM routing. Predicts model utilities by averaging over the k nearest training queries in image-feature space. This baseline supports arbitrary cost functions at inference.

Linear regression (LR). A linear baseline predicting model utilities from image features, also studied by Li [30]. This baseline supports arbitrary cost functions at inference.

SCOUT without decoupling. SCOUT trained without the score decoupling procedure. Specifically, the softmax target is over the full score vector $\mathbf{s}^{(i)}$ including view-invariant scores, rather than over $\mathbf{s}_{\text{dep}}^{(i)}$ alone. This baseline isolates the contribution of the decoupling procedure as opposed to the rest of the SCOUT framework.

Among these baselines, ZOOTHER is the closest to SCOUT in design philosophy. The key methodological difference is SCOUT’s decoupling, which better captures the heterogeneous model pool and allows for arbitrary cost functions at inference. ZOOTHER extended to support arbitrary cost functions through utility optimization is equivalent to SCOUT without decoupling.

5.1.7 Evaluation Metrics

Following standard practice in routing evaluation [28,30], we report *regret* as defined in Equation 3.1: the expected utility gap between the router’s selection and the oracle’s selection, averaged over a held-out evaluation set. Lower regret indicates a better router. We report all regrets as a proportion of the input-agnostic baseline’s regret on the same cost regime and metric, which normalizes across datasets and mesh quality metrics.

For multi-dimensional cost families, we evaluate over a large number of cost functions that sweep over the cost family and report the average regret. For one-dimensional cost families, we sweep λ over a range and report two summary metrics adapted from RouterBench [28]: the *deferral curve*, which plots achieved utility against total cost allocation; and the *Average Improvement in Quality (AIQ)*. By sweeping λ , we obtain different achieved utilities at different total cost allocations, from which we can construct a Pareto frontier ($Q_h(c)$) as shown in Fig. 5.3 which represents the deferral curve. The AIQ is then defined as:

$$\text{AIQ}(h) = \frac{1}{c_{\max} - c_{\min}} \int_{c_{\min}}^{c_{\max}} Q_h \, dc.$$

For these metrics prescribed by RouterBench, we additionally include the Zero router baseline [28], which randomly samples routers at a given total cost allocation to trace out the non-decreasing convex hull.

5.1.8 Implementation Details

Probability network $\hat{\mathbf{p}}$. 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 a batch size 128 for 15 epochs. The same architecture is used for all baselines that learn an MLP, ensuring a fair comparison.

Tag-based smoothing and softmax temperature. For GSO, we use $\beta = 0.7$ and a softmax temperature $T = 1.0$. For BigBIRD + YCB, the scores are noisier (smaller dataset, real-image and rendering variation), so we decrease β to 0.5 for stronger smoothing toward category averages. The softmax temperature T is set per metric to account for differences in score scale; the per-metric values for BigBIRD + YCB are listed in Table 5.2.

Table 5.2: 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	0.1	5.0	0.15	0.1	1.0

Partition function network $\hat{z}$. Predicted by ridge regression with regularization parameter $\alpha = 10^3$. The weighting terms R_j^2 , S_j^2 , and the expectation in w_j from Equation 4.1 are estimated via 5-fold cross-validation over the training set.

Image features. For both networks (and all baselines), input images are encoded by concatenating features from four pretrained vision encoders: CLIP ViT-B/32 (512-d) [47], ResNet-50 (2048-d) [48], ViT-B/16 (768-d) [49], and ConvNeXt-B (1024-d) [50], yielding a 4,352-dimensional input.

Cost values. Latency costs (in seconds): 30 (Hunyuan3D), 10 (InstantMesh), 10 (TRELLIS), 0.5 (TripoSR), 654 (2DGS), 654 (NeRF2Mesh), 330 (NeuS2). Memory costs (in GB): 6 (Hunyuan3D), 24 (InstantMesh), 16 (TRELLIS), 6 (TripoSR). We omit view-invariant methods from the memory-based cost families because their inference-time memory usage differs structurally from Image-to-3D methods (they operate on multi-view captures rather than producing a single forward pass), making direct comparison less meaningful. The *skip* option is also excluded from one-dimensional 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 [51–54] when available. Otherwise, they are approximated by experiments, where we observed similar latency between NeRF2Mesh and 2DGS so we approximated the NeRF2Mesh latency to equal the latency reported from 2DGS.

Data collection. Ground-truth meshes are obtained from the GSO and YCB datasets. For GSO, images are rendered at elevation angles $\{35^\circ, 45^\circ, 60^\circ, 75^\circ, 85^\circ\}$ with three azimuth angles $\{30^\circ, 120^\circ, 210^\circ\}$ per elevation, producing nondegenerate views. For YCB, images are captured at elevation angles $\{5^\circ, 22.5^\circ, 45^\circ, 67.5^\circ, 90^\circ\}$ with azimuth angles $\{0^\circ, 45^\circ, 90^\circ\}$ per the first four elevations and a single top view at azimuth 45° for the 90° elevation, producing many degenerate views. Each YCB viewpoint is captured under three conditions: flash-style lighting, surround-style lighting, and a real photograph from the BigBIRD dataset. Examples of degenerate versus nondegenerate views are shown in Figure 5.1 and examples of different lighting styles are shown in Figure 5.2. 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.

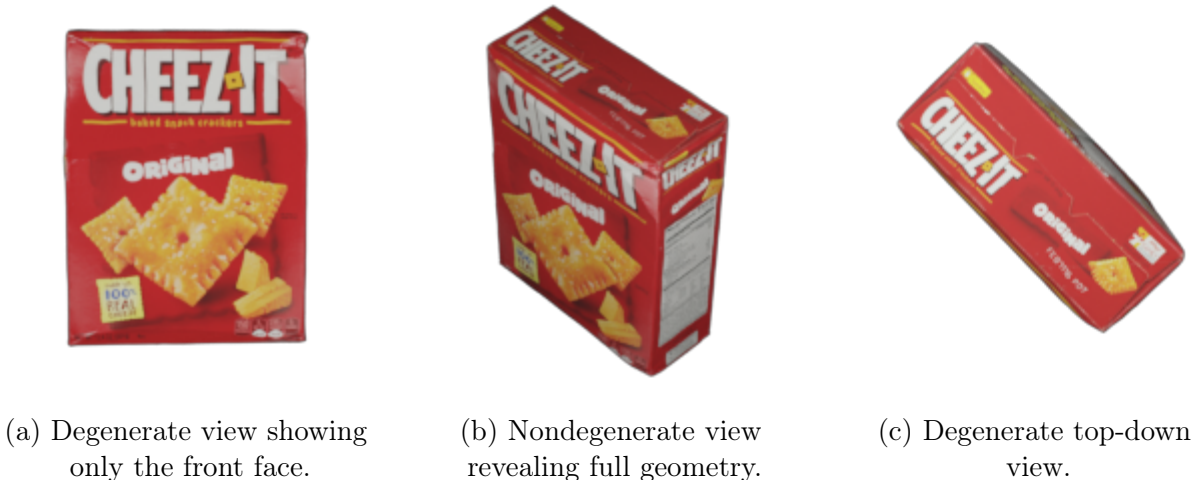


Figure 5.1: Degenerate and nondegenerate views of a cracker box.

For Hunyuan3D, we apply a convex decomposition using CoACD [55] (concavity threshold 0.025, 50 MCTS search iterations) between the mesh and texture generation stages to prevent



(a) Rendered image with flash-style lighting.

(b) Rendered image with surround-style lighting.

(c) Real image from BigBIRD [32].

Figure 5.2: Different lighting conditions for a fixed viewpoint.

the texturing process from taking excessive time. For each of the three view-invariant models (2DGS, NeRF2Mesh, NeuS2), we generate a single mesh per object using 120 images captured at elevation angles $\{35^\circ, 45^\circ, 60^\circ, 75^\circ, 85^\circ\}$ with 24 azimuth angles spanning 360° per elevation. We observed minimal difference in reconstruction quality between using 120 images versus smaller numbers such as 24. The *skip* option requires no reconstruction since it has a fixed default score. We use default parameters for all reconstruction pipelines.

Mesh registration and scoring. The ground-truth and reconstructed meshes are normalized to a common scale by fitting each to a unit sphere centered at the origin using *miniball* [56]. We then register each reconstructed mesh to its ground-truth counterpart via FoundationPose [57] for global registration, followed by DDM [58] for local refinement (learning rate 2×10^{-2} , 100 iterations). The seven mesh quality metrics are then computed on the registered meshes.

Compute and seeds. 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.

5.2 Routing Performance on Benchmark Datasets

We evaluate SCOUT against the baselines from Section 5.1.6 across the cost regimes and metrics defined in Sections 5.1.5 and 5.1.3. The evaluation is structured to stress different aspects of the routing problem: regret across cost regimes on GSO, regret across quality metrics on BigBIRD + YCB, and routing under one-dimensional cost families that mirror the LLM routing setting. We then summarize the comparison with the adapted LLM routing baselines.

5.2.1 Regret Across Cost Families on GSO

Table 5.3 reports regret on novel objects across four of the cost regimes defined in Section 5.1.5, with reconstruction quality measured by DCD. These regimes span the full range of routing scenarios: pure quality-based selection ($c_{\text{no-cost}}$), cost-aware routing over the full model pool ($c \in \mathcal{C}$), and the two viewpoint-dependent-only regimes.

Table 5.3: Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret across different cost regimes on the GSO dataset, evaluated by DCD. SCOUT no dc refers to SCOUT (ours) without decoupling.

Method	$c \in \mathcal{C}$	$c \in \mathcal{C}_{\text{dep}}$	$c_{\text{no-cost}}$	$c_{\text{view, no-cost}}$
ZOOTER	N/A	N/A	0.6580 ± 0.0050	0.6769 ± 0.0039
RouterDC	N/A	N/A	0.8013 ± 0.0050	1.0200 ± 0.0073
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	0.6261 ± 0.0049	0.6489 ± 0.0033
LR	0.4909 ± 0.0019	0.5046 ± 0.0019	0.6361 ± 0.0046	0.6497 ± 0.0033
SCOUT no dc	0.5249 ± 0.0021	0.5146 ± 0.0019	0.6580 ± 0.0050	0.6769 ± 0.0039
SCOUT (ours)	0.4734 ± 0.0019	0.4810 ± 0.0019	0.6340 ± 0.0046	0.6282 ± 0.0035
Input-agnostic	1.0000	1.0000	1.0000	1.0000

SCOUT achieves statistically significantly lower regret than all baselines in three of the four cost regimes, and is within the kNN’s error bar on the fourth ($c_{\text{no-cost}}$).

The MLP and MF baselines, both adapted from LLM routing (RouterBench and RouteLLM respectively), perform substantially worse than the simpler kNN and LR baselines on this task. This mirrors the finding from Li [30] in the LLM routing setting: simple non-parametric and linear methods often outperform more elaborate learned routers. SCOUT and ZOOTER both perform well, even though they are parameterized models. We conjecture that this arises because the KL divergence loss is more suitable for the routing problem, as it offsets the input-specific difficulty and instead focuses on learning the relative rankings between models.

5.2.2 Regret Across Quality Metrics on BigBIRD + YCB

Different downstream manipulation tasks require different reconstruction quality criteria, as discussed in Section 5.1.3. A useful router should be able to learn from a dataset based on scores that represent any metric rather than overfitting to a single notion of quality. Table 5.4 reports regret on BigBIRD + YCB at $c \in \mathcal{C}$ across all seven quality metrics, with the ZOOTER and RouterDC rows omitted as they do not support this cost regime.

SCOUT achieves the lowest regret on five of the seven metrics (DCD, IoU, MMD-EMD, Eval3D-geo, Eval3D-struct), often by substantial margins—notably halving the regret of the closest baseline on Eval3D-geo. The two Chamfer metrics (Chamfer L1, Chamfer L2) are the exceptions. For these metrics, all learned routers, including SCOUT, fail to beat the input-agnostic baseline. This is a consequence of how view-invariant methods produce reconstructions. 2DGS, NeRF2Mesh, and NeuS2 generate meshes from images that never

Table 5.4: Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret for $c \in \mathcal{C}$ across multiple quality metrics on BigBIRD + YCB. SCOUT no dc denotes SCOUT (ours) without decoupling.

Method	DCD	Chamfer L2	Chamfer L1	IoU	MMD-EMD	Eval3D-geo	Eval3D-struct
MLP	1.0303 ± 0.0165	15.2150 ± 2.7605	1.7054 ± 0.1702	0.7681 ± 0.0166	1.0942 ± 0.0514	0.8845 ± 0.0373	0.8905 ± 0.0132
MF	1.2078 ± 0.0175	13.1332 ± 1.6315	2.2107 ± 0.1355	0.9076 ± 0.0149	1.9530 ± 0.0572	2.7425 ± 0.1694	1.3766 ± 0.0319
kNN	0.9282 ± 0.0167	1.7599 ± 0.2038	1.0074 ± 0.0779	0.7474 ± 0.0143	0.9788 ± 0.0438	1.1768 ± 0.1278	0.8522 ± 0.0185
LR	0.8919 ± 0.0169	7.8104 ± 1.2446	1.4418 ± 0.1288	0.7330 ± 0.0142	1.0073 ± 0.0494	0.7511 ± 0.0384	0.8258 ± 0.0212
SCOUT no dc	0.9258 ± 0.0155	10.8840 ± 1.9624	1.3903 ± 0.1548	0.6837 ± 0.0135	0.9678 ± 0.0502	0.6893 ± 0.0572	0.8587 ± 0.0274
SCOUT (ours)	0.8489 ± 0.0164	6.6110 ± 0.9977	1.3258 ± 0.1482	0.6248 ± 0.0148	0.9146 ± 0.0358	0.3547 ± 0.0109	0.8243 ± 0.0224
Input-agnostic	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

show the bottom of the object, since the supporting surface occludes any view from below. Consequently, these view-invariant methods produce hollow meshes that often contain interior artifacts with irregular surfaces that yield extremely noisy Chamfer scores.

Across the remaining five metrics, SCOUT consistently achieves the lowest regret, demonstrating that it is robust to training and evaluation across different quality metrics.

5.2.3 One-Dimensional Cost Families

The LLM routing literature evaluates routers using one-dimensional cost families via metrics including the deferral curve and Average Improvement in Quality (AIQ), as discussed in Section 5.1.7. We evaluate SCOUT under the same setting to enable direct comparison with the conventions established by RouterBench [28].

We use the three cost families defined in Section 5.1.5: $\mathcal{C}_{\text{latency}}$, $\mathcal{C}_{\text{memory}}$, and $\mathcal{C}_{\text{latency} \times \text{memory}}$, penalizing inference latency, peak memory, and memory-time occupancy (GB·s) respectively, each parameterized by a scalar $\lambda \geq 0$ swept over a range of values. ZOOTOER and RouterDC are excluded as they do not support varying λ .

Figure 5.3 shows the deferral curves for the three cost families: $\mathcal{C}_{\text{latency}}$ (left), $\mathcal{C}_{\text{memory}}$ (middle), and $\mathcal{C}_{\text{latency} \times \text{memory}}$ (right) and Table 5.5 reports AIQ for each method on each cost family, evaluated on novel GSO objects. SCOUT achieves the highest AIQ across all three cost families. The magnitude of improvement is consistent with the inter-method differences reported by RouterBench [28] in the LLM routing setting, where gains between routing methods are similarly modest.

Table 5.5: AIQ ($\uparrow$) for one-dimensional cost families on the GSO dataset, evaluated on novel objects.

Method	Latency	Memory	Latency×Memory
MLP	0.5730 ± 0.0010	0.8158 ± 0.0007	0.7756 ± 0.0007
MF	0.5661 ± 0.0012	0.8138 ± 0.0012	0.7722 ± 0.0012
kNN	0.5772 ± 0.0010	0.8307 ± 0.0006	0.7847 ± 0.0007
LR	0.5774 ± 0.0010	0.8284 ± 0.0006	0.7832 ± 0.0007
SCOUT no dc	0.5763 ± 0.0010	0.8275 ± 0.0006	0.7826 ± 0.0007
SCOUT (ours)	0.5801 ± 0.0010	0.8348 ± 0.0006	0.7887 ± 0.0007
Zero router	0.5539 ± 0.0008	0.7126 ± 0.0008	0.6914 ± 0.0006
Oracle	0.6639 ± 0.0009	0.9459 ± 0.0003	0.8813 ± 0.0005

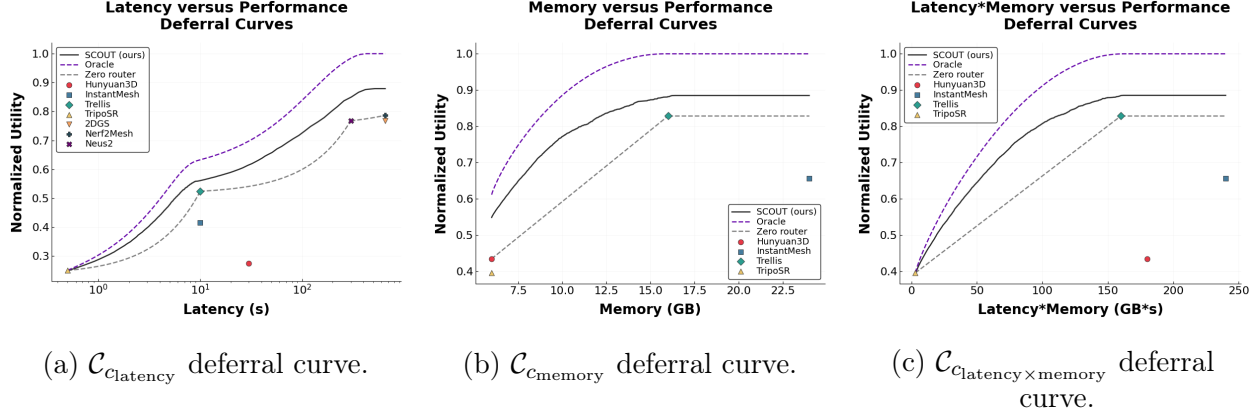


Figure 5.3: Deferral curves for the three cost families $\mathcal{C}_{\text{latency}}$ (left), $\mathcal{C}_{\text{memory}}$ (middle), and $\mathcal{C}_{\text{latency} \times \text{memory}}$ (right), showing the normalized utility achieved by each routing method as a function of the imposed cost constraint.

5.2.4 Comparison with Adapted LLM Routing Baselines

Across the three regimes evaluated above, several patterns emerge in the comparison between SCOUT and the LLM-routing-derived baselines.

ZOOTER and RouterDC are limited to a single fixed cost function. Both methods learn a fixed per-input model preference during training and cannot recombine that preference under arbitrary inference-time costs.

MLP and MF underperform simpler baselines. The neural baselines from RouterBench (MLP) and RouteLLM (MF) perform worse than kNN and LR across most cost families and metrics. This mirrors the finding from Li [30] in LLM routing, and confirms that the same pattern holds for 3D reconstruction routing.

kNN and LR are SCOUT’s closest competitors. Across both GSO and BigBIRD + YCB, the strongest baselines are consistently the simple kNN and LR methods. SCOUT outperforms both, achieving under half the regret of these baselines for some experiments. SCOUT is also more efficient: end-to-end training and evaluation on GSO 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 since SCOUT’s runtime is effectively independent of k_1 , while kNN and LR scale linearly—an advantage that increases as new Image-to-3D models emerge.

SCOUT without decoupling consistently underperforms full SCOUT and often matches or trails the simpler kNN and LR baselines. The decoupling procedure additionally enables for strong performance even as more view-invariant methods are added, as seen in Figure 4.2.

5.3 Robotic Manipulation Validation

The benchmark routing results in Section 5.2 establish that SCOUT achieves lower regret than adapted LLM routing baselines on standard mesh quality metrics. This section validates that the routing improvements translate to measurable gains on downstream robotic manipulation tasks, where the ultimate utility of a reconstruction is determined not by mesh-quality scores

but by whether the robot can actually use the reconstruction to complete a task. We evaluate across three tasks of increasing complexity: grasp proposal evaluation against ground-truth meshes (Section 5.3.1), dexterous manipulation in simulation (Section 5.3.2), and real-world pick-and-place on a Franka Panda robot (Section 5.3.3). Each task is conducted following the experimental protocol of SceneComplete [17].

5.3.1 Grasp Proposal Evaluation

We first evaluate whether SCOUT’s reconstructions support reliable grasp planning. Specifically, we generate two-fingered antipodal grasp proposals [59] on each reconstructed mesh and report two metrics against the ground-truth mesh:

- Grasp collision rate, the fraction of proposed grasps whose gripper geometry intersects the ground-truth mesh (lower is better)
- Mesh-IoU, the volumetric overlap between the reconstructed and ground-truth meshes (higher is better)

We evaluate on 10 YCB objects representing a diverse range of geometries.

Table 5.6 reports per-object and mean results. SCOUT is evaluated under two settings: novel views of known objects (the object was seen from other viewpoints during training) and novel views of novel objects (the object was not seen during training at all). The latter is the more demanding setting and the one most representative of deployment, where a robot must handle objects it has not been trained on.

Table 5.6: Grasp collision rate ($\downarrow$) / mesh-IoU ($\uparrow$) across 10 YCB objects, using a two-fingered gripper. SCOUT is evaluated under two settings: routing over novel views of known objects, and over novel views and novel objects.

Object	Hunyuan3D	InstantMesh	TRELLIS	TripoSR	SCOUT (novel obj.)	SCOUT (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

No single model dominates across objects. Specifically, examining the per-object results, the best-performing reconstruction model varies by object. For example, InstantMesh creates a mesh with near-zero grasp collisions and 0.9164 mesh-IoU on the tomato soup can but degrades to a 0.6098 fraction of collisions on the potted meat container. On the other hand, Hunyuan3D produces a mesh with zero grasp collisions for both cups, but many collisions and a low IoU for the tomato soup can. TRELLIS is competitive on the chef can but degrades on the brick. This object-dependent variation in model performance is exactly what motivates per-input routing: a fixed-model pipeline inherits the worst-case failure modes of its chosen

model, while a router can avoid those failures by selecting alternative models for the inputs where they would occur.

Further, SCOUT outperforms choosing any specific, single reconstruction model. On both the novel-object and novel-view settings, SCOUT achieves the lowest mean collision rate (0.0729 and 0.0251 respectively) and the highest mean mesh-IoU (0.6278 and 0.6854). The gap is substantial: the best single-model baseline on grasp collision rate is Hunyuan3D at 0.0939, while the best on mesh-IoU is also Hunyuan3D at 0.5296. SCOUT improves on the best single-model grasp collision rate by 22% on novel objects and 73% on novel views, with corresponding gains in mesh-IoU. This confirms that per-input routing translates directly into measurable downstream gains as the router selects the appropriate model for each object, avoiding the failure modes that a fixed-model pipeline would inherit.

The right two columns of Table 5.6 show the model that SCOUT actually selected for each object. The selections track the per-object best-performer pattern visible in the left columns: SCOUT selects InstantMesh for cans, Hunyuan3D for cups, and TripoSR for the brick. Each of these selections minimizes the grasp collision rates.

Figure 5.4 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 how per-input routing can simultaneously improve reconstruction quality and reduce computational cost relative to fixed-model alternatives. Figures 5.5 and 5.6 show the eight remaining objects from Table 5.6. Across these objects, SCOUT routes to a variety of reconstruction models depending on the input viewpoint and object, confirming that no single model dominates and that per-input selection is beneficial.

5.3.2 Dexterous Manipulation in Simulation

While two-fingered grasping requires only that a reconstructed mesh approximate the object’s outer surface well enough to avoid collisions, dexterous manipulation imposes substantially stricter requirements. Multi-finger contact placement and force closure depend on local surface geometry. Consequently, small errors in surface normal or curvature can cause the grasp to slip or apply incorrect forces. Differences between reconstruction models are therefore amplified relative to two-fingered grasping, providing a more sensitive test of routing.

We evaluate dexterous manipulation success rate in simulation across five YCB objects, using a multi-finger gripper with contact-aware grasp synthesis. Table 5.7 reports the success rate (% of attempts ending with the object grasped and lifted) for each reconstruction model on each object, along with SCOUT’s selected model and the resulting mean success rate. By routing per object, SCOUT achieves a mean success rate of 41.6%, outperforming the best single-model baseline (Hunyuan3D, 35.6%).

5.3.3 Real-World Pick-and-Place on a Franka Panda

To validate the full pipeline end-to-end on a physical robot, we evaluate real-world pick-and-place success on a Franka Panda robot. For each test object, we capture a single RGB image, generate a reconstructed mesh using each Image-to-3D model, sample antipodal grasp

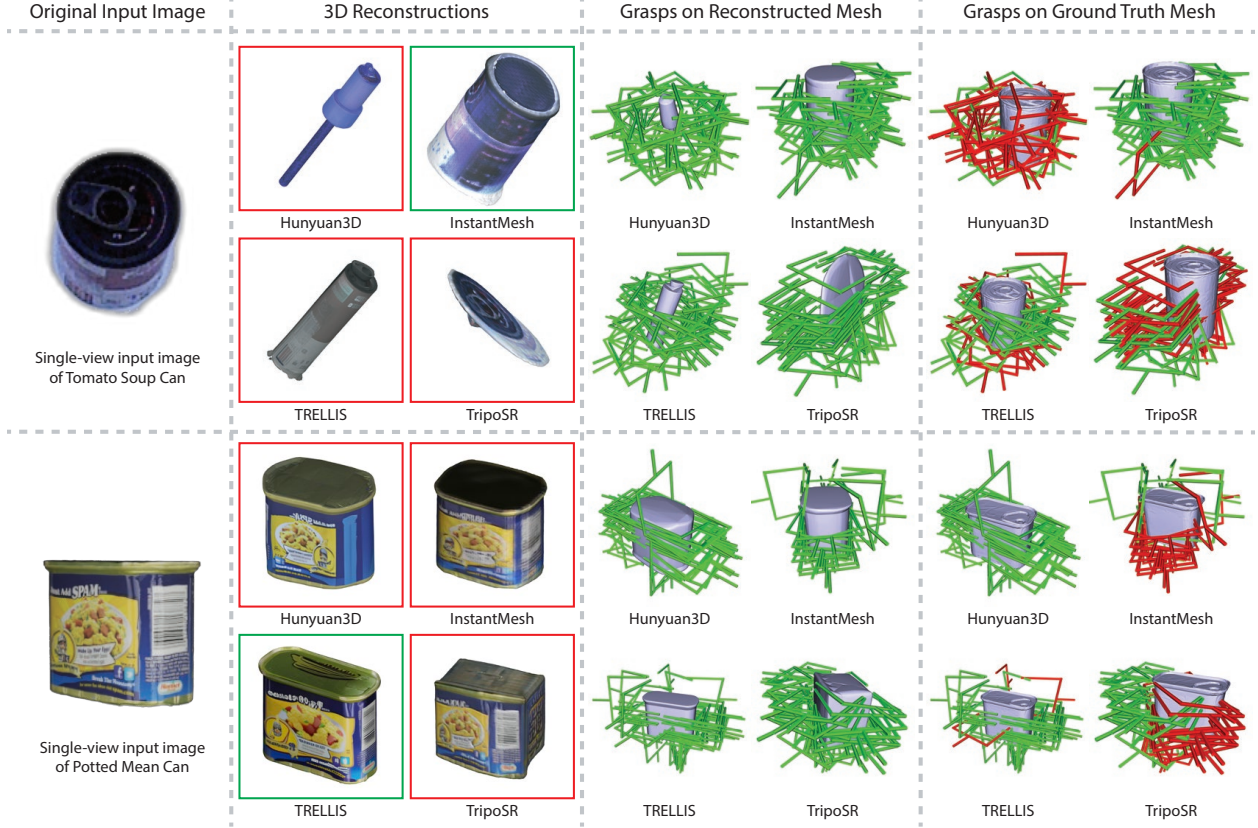


Figure 5.4: Evaluation of grasp proposals on reconstructed meshes for two example inputs. From left to right: (1) original input image to SCOUT, (2) 3D reconstructions from each model with the model chosen by SCOUT boxed in green, (3) grasp proposals on the reconstructed mesh, (4) grasp proposals evaluated on the ground-truth mesh; colliding grasps are shown in red. SCOUT selects InstantMesh for the top-down can view (where it is the only model producing a usable reconstruction) and TRELIS for the frontal meat container view (where it reconstructs accurately at lower latency than Hunyuan3D).

proposals [59] on the reconstructed mesh, and execute 5 sampled grasps per object on the Franka Panda robot. A grasp is considered successful if the object is lifted and held without slipping. Table 5.8 reports the per-model and SCOUT-selected results across five objects.

Real-world deployment confirms the routing benefit. SCOUT achieves a mean success rate of 4.6/5, outperforming every single-model baseline as seen in Table 5.8. The improvement is small in absolute terms but reflects two structural facts about the deployment setting. First, on the inputs where Hunyuan3D succeeds, no router can do better. Rather, the gain comes from inputs where Hunyuan3D underperforms relative to an alternative. Second, the router’s selections reflect input difficulty. SCOUT chooses Hunyuan3D on challenging viewpoints where it produces the highest-fidelity reconstructions and TRELIS on easier viewpoints where TRELIS suffices and offers lower latency. The router thus delivers higher mean performance and lower expected latency.

Table 5.7: Dexterous manipulation success rate ($\uparrow$) in simulation for each reconstruction model across 5 objects (HY3D: Hunyuan3D, IM: InstantMesh, TRL: TRELIS, TSR: TripoSR). **SCOUT (ours)** reports the model selected by SCOUT for each object and the resulting mean success rate.

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
SCOUT (ours)	TRL	HY3D	IM	TSR	HY3D	41.6

Table 5.8: Real-world pick-and-place success rate ($\uparrow$) on a Franka Panda robot across 5 objects. For each reconstructed object, 5 antipodal grasps are sampled [59] and executed. **SCOUT (ours)** reports the model selected by SCOUT for each object and the resulting mean success rate.

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
SCOUT (ours)	HY3D	TRL	HY3D	HY3D	TRL	4.6/5

5.4 Design Choice Ablations

The previous section established that SCOUT outperforms adapted LLM routing baselines across cost function regimes, quality metrics, and one-dimensional cost families. This section isolates the contribution of each design choice in SCOUT through controlled ablations: the tag-based smoothing of training targets (Section 5.4.1), the weighted proxy for the partition function (Section 5.4.2), and the sensitivity of routing performance to hyperparameter choices (Section 5.4.3).

5.4.1 Effect of Smoothing

SCOUT applies tag-based smoothing to the training targets for $\hat{\mathbf{p}}$, blending each per-image score with the mean score over training examples sharing the same tag. The smoothing reduces noise in the relative-performance training signal as discussed in Chapter 4. Table 5.9 reports SCOUT’s regret with and without smoothing across the $c_{\text{view, no-cost}}$ regime. This subspace was chosen for comparison as it measures accuracy of the viewpoint-dependent methods without considering the view-invariant methods, and the main effect of smoothing is for the viewpoint-independent methods. This ablation demonstrates a deterioration in SCOUT’s performance when we remove the smoothing when learning the $\hat{\mathbf{p}}$ model.

Smoothing helps relative-performance prediction but hurts absolute-score prediction. The asymmetric application of smoothing in SCOUT—applied when training $\hat{\mathbf{p}}$, but not

Table 5.9: Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret across the $c_{\text{view, no-cost}}$ regime on GSO, with and without tag-based smoothing. Lowest regret per method is in **bold**.

Method	SCOUT (ours)	LR	kNN
With tag-based smoothing	0.6282 ± 0.0035	0.6611 ± 0.0035	0.6513 ± 0.0034
Without tag-based smoothing	0.6338 ± 0.0034	0.6497 ± 0.0033	0.6489 ± 0.0033

applied when constructing proxy targets for $\hat{z}$ —is justified by the differing roles of the two networks. The probability network $\hat{\mathbf{p}}$ benefits from smoothed targets because it only aims to learn the relative performance of the models, rather than absolute scores which include the per-image difficulty. The partition function network $\hat{z}$, by contrast, must capture per-image difficulty—which smoothing actively suppresses by blending toward category averages.

To validate this reasoning, we apply smoothing to two baselines that, like $\hat{z}$, predict absolute scores rather than relative probabilities: kNN and LR. If smoothing genuinely interferes with absolute-score prediction, we should observe degradation rather than improvement in these baselines. Table 5.9 confirms this as smoothing harms both LR and kNN performance, in contrast to the improvement observed in SCOUT.

5.4.2 Weighted Proxy Comparison

SCOUT learns the partition function $\hat{z}$ by regressing on a weighted average of k_1 per-model proxies, with weights derived in Chapter 4 via inverse-variance optimization. We compare the derived weighting against three natural alternatives:

- **Ground-truth $z^{(i)}$.** Train $\hat{z}$ to regress directly on the true partition function $z^{(i)} = \sum_{m=1}^{k_1} e^{s_{\text{dep},m}^{(i)}/T}$. This is the apparent “correct” target.
- **Equal weighting.** Set $w_j = 1/k_1$ for all j , treating each per-model proxy as equally informative.
- **One-hot weighting.** Set $\tilde{z}^{(i)} = \tilde{z}_{j^*}^{(i)}$, where $j^* = \arg \max_j w_j$ is the model with the largest derived weight.

We compare these alternatives on BigBIRD + YCB, where the smaller dataset and noisier scores stress the partition function estimation more than on GSO. Table 5.10 reports the regret under each of the four alternative target choices for learning the $\hat{z}$ model. SCOUT’s derived weighting achieves lower regret than each alternative, evaluated across cost functions sampled from $\mathcal{C}$.

The weighted proxy improves over all three alternatives on the majority of metrics. The most striking result is the comparison against the ground-truth $z^{(i)}$ where the weighted proxy outperforms the apparent “correct” target on all seven metrics. This is initially counterintuitive—one might expect that regressing on the true target produces the best estimator. The explanation lies in the variance of the partition function. 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

Table 5.10: Regret ($\downarrow$) as a proportion of the input-agnostic baseline regret for $c \in \mathcal{C}$ across multiple quality metrics on BigBIRD + YCB. SCOUT uses the derived weighted average from Equation (4.1); True z uses the exact partition function; Equiweighted uses a uniform average across models; One-hot uses a single model. Lowest regret per metric is in **bold**.

Metric	SCOUT	True z	Equiweighted	One-hot
DCD	0.8489 $\pm$ 0.0164	0.9004 $\pm$ 0.0148	0.8981 $\pm$ 0.0150	0.9081 $\pm$ 0.0153
Chamfer L2	6.6110 $\pm$ 0.9977	7.2530 $\pm$ 1.0896	7.3726 $\pm$ 1.1114	6.5034 $\pm$ 0.9822
Chamfer L1	1.3258 $\pm$ 0.1482	1.4076 $\pm$ 0.1179	1.4168 $\pm$ 0.1202	1.3708 $\pm$ 0.1143
IoU	0.6248 $\pm$ 0.0148	0.7280 $\pm$ 0.0135	0.7281 $\pm$ 0.0135	0.7284 $\pm$ 0.0142
MMD-EMD	0.9146 $\pm$ 0.0358	0.9148 $\pm$ 0.0354	0.9156 $\pm$ 0.0358	0.9214 $\pm$ 0.0352
Eval3D-geo	0.3547 $\pm$ 0.0109	0.8636 $\pm$ 0.0452	1.2387 $\pm$ 0.0692	0.7372 $\pm$ 0.0395
Eval3D-struct	0.8243 $\pm$ 0.0224	0.8359 $\pm$ 0.0208	0.8363 $\pm$ 0.0207	0.8347 $\pm$ 0.0205

downweighting model j when (i) its predicted probabilities $\hat{p}_j$ are noisier (large $S_j^2(1 - R_j^2)$), or (ii) its true probability is frequently low, making the proxy $\hat{z}_j^{(i)}$ a higher-variance estimate of $z^{(i)}$ (see Equation 4.5).

The comparison against equal weighting confirms that the inverse-variance scheme captures meaningful structure: simply averaging the proxies discards information about which models contribute reliably and which contribute noise. The comparison against one-hot weighting confirms that combining multiple proxies—rather than relying on the single best one—provides variance reduction. Taken together, these comparisons validate the inverse-variance derivation in Chapter 4 as the appropriate weighting choice for the partition function estimation problem.

5.4.3 Hyperparameter Sensitivity

An effective routing framework should have a relatively well-shaped hyperparameter landscape. We verify this by performing one-at-a-time sweeps over each hyperparameter for SCOUT, holding the others fixed. These experiments were conducted on seeds 0–49 and evaluated on novel objects in the GSO dataset.

SCOUT hyperparameters. The selected hyperparameters for GSO are $\beta = 0.7$ (smoothing), learning rate 10^{-4} , 15 training epochs, $T = 1.0$ (softmax temperature), and $\alpha = 10^3$ (ridge regression regularization parameter). β , the learning rate, and the number of epochs only affect $\hat{\mathbf{p}}$, whereas α only affects the $\hat{z}$ model. The temperature T has effects on both, as it is used in the label distribution to learn $\hat{\mathbf{p}}$ and in the equation for weighted proxy labels to learn $\hat{z}$. We therefore performed hyperparameter sweeps under the cost function $c_{\text{view, no-cost}}$ for β , the learning rate, and the number of epochs, and under the cost-function family $\mathcal{C}$ for T and α . The hyperparameter landscapes are shown in Figures 5.7 and 5.8. Regret varies smoothly across each of the five hyperparameters, with a clear local optimum near our selected value, and regret increases gradually as the hyperparameter moves away from the optimum.

Baseline hyperparameters. Since Li [30] and our experiments both find that kNN and LR are the strongest baselines, we also analyze their hyperparameter sensitivity, sweeping k for the kNN and the regularization parameter α for the LR model. Both exhibit smooth

landscapes (Figure 5.9), confirming that the comparison in Section 5.2 is not an artifact of misspecified baseline hyperparameters.

5.5 What SCOUT Learns

The previous sections establish that SCOUT outperforms baselines on benchmark routing metrics and downstream manipulation tasks. We now examine what the router has actually learned by examining the policy mapping inputs to model selections. First, we explore the per-category routing behavior which reveals whether SCOUT has learned that different reconstruction models are appropriate for different object types. Second, we explore the per-viewpoint routing behavior which reveals whether SCOUT has learned that different reconstruction models are appropriate for different viewing angles. Both axes are evaluated on novel objects, testing whether the learned policy generalizes beyond the training distribution rather than memorizing per-object preferences.

5.5.1 Per-Category Routing Behavior

We compare SCOUT’s learned routing policy against the oracle policy across object categories. The oracle policy, computed by selecting the best model for each input given access to the ground-truth scores, captures the true category-conditional distribution of which model is optimal. The learned policy captures what SCOUT actually selects. If the two distributions agree, SCOUT has learned the category-dependent biases of the underlying reconstruction models from the training data and applies them correctly to novel objects.

Figure 5.10 shows the oracle and learned distributions side by side: each row is an object category, each column is a reconstruction model, and the cell value is the fraction of inputs in that category for which the corresponding model is selected. The oracle (Figure 10(a)) reveals strong category-dependent preferences: certain categories are dominated by one or two models, while others spread the selection more uniformly. The learned policy (Figure 10(b)) reproduces these preferences closely. Notably, the learned policy does bias TRELLIS in all categories as that is the overall best model for the GSO dataset, yet SCOUT’s policy successfully captures the trends seen in the oracle. Categories where the oracle concentrates on a single model show a similar concentration in the learned policy whereas categories where the oracle distributes selection across multiple models show a similar trend in the learned policy. This agreement is achieved despite SCOUT being evaluated on objects it has not seen during training, indicating that the category-dependent preferences are a genuine property of the input distribution rather than a per-object memorization.

5.5.2 Per-Viewpoint Routing Behavior

The second axis along which SCOUT may learn structure is viewpoint. Figure 5.11 shows the oracle and learned policies grouped by elevation: high elevations (75° and 85°) approach top-down views, while low elevations (35° , 45° , 60°) approach more side views. The oracle distribution (Figure 11(a)) shows viewpoint-dependent structure: TripoSR vastly underperforms at high elevation angles (only optimal 6.33% of the time) and is relatively better on

the low elevation angles (optimal 18.12% of the time). The opposite bias can be observed for InstantMesh and TRELLIS, which both perform better on the high elevation angles. The learned policy (Figure 11(b)) reproduces this structure, with the same exception as in Section 5.5.1 that the learned policy biases toward TRELLIS across both viewpoints, since TRELLIS is the overall best model for the GSO dataset.

5.5.3 Implications

The agreement between oracle and learned policies on both axes confirms two properties of SCOUT. First, the selections of the learned router reflect interpretable structure—category- and viewpoint-dependent model preferences—that aligns with the underlying biases of the reconstruction models. Second, this structure generalizes to novel objects, indicating that SCOUT has learned distributional properties of the input space rather than per-instance preferences. The robotic manipulation results in Section 5.3 are a direct consequence: SCOUT’s per-input selections are guided by genuine, generalizable structure in the input distribution, and that structure translates into measurable downstream gains.

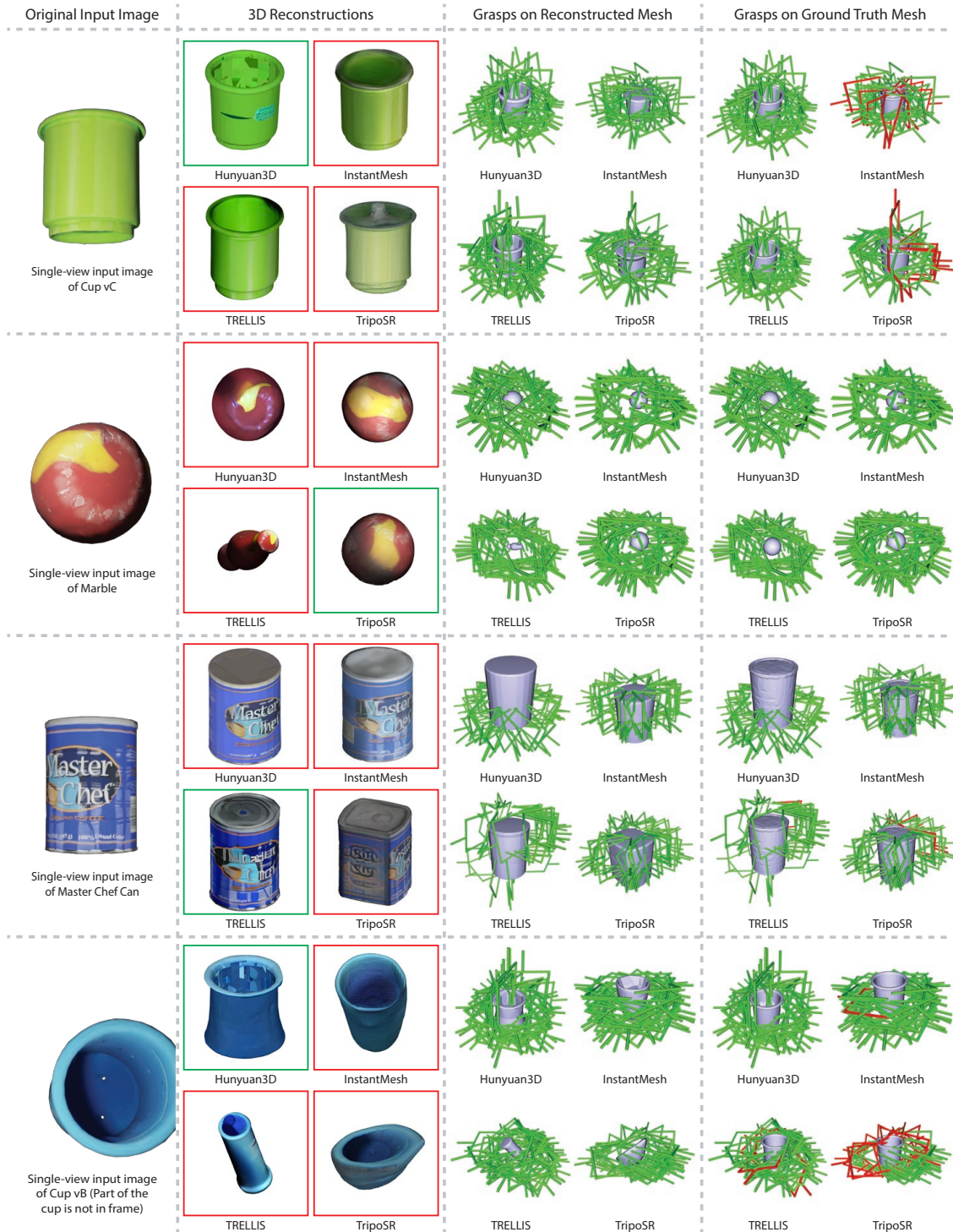


Figure 5.5: Evaluation of grasp proposals on the first half of the remaining eight YCB objects. Layout follows Figure 5.4.

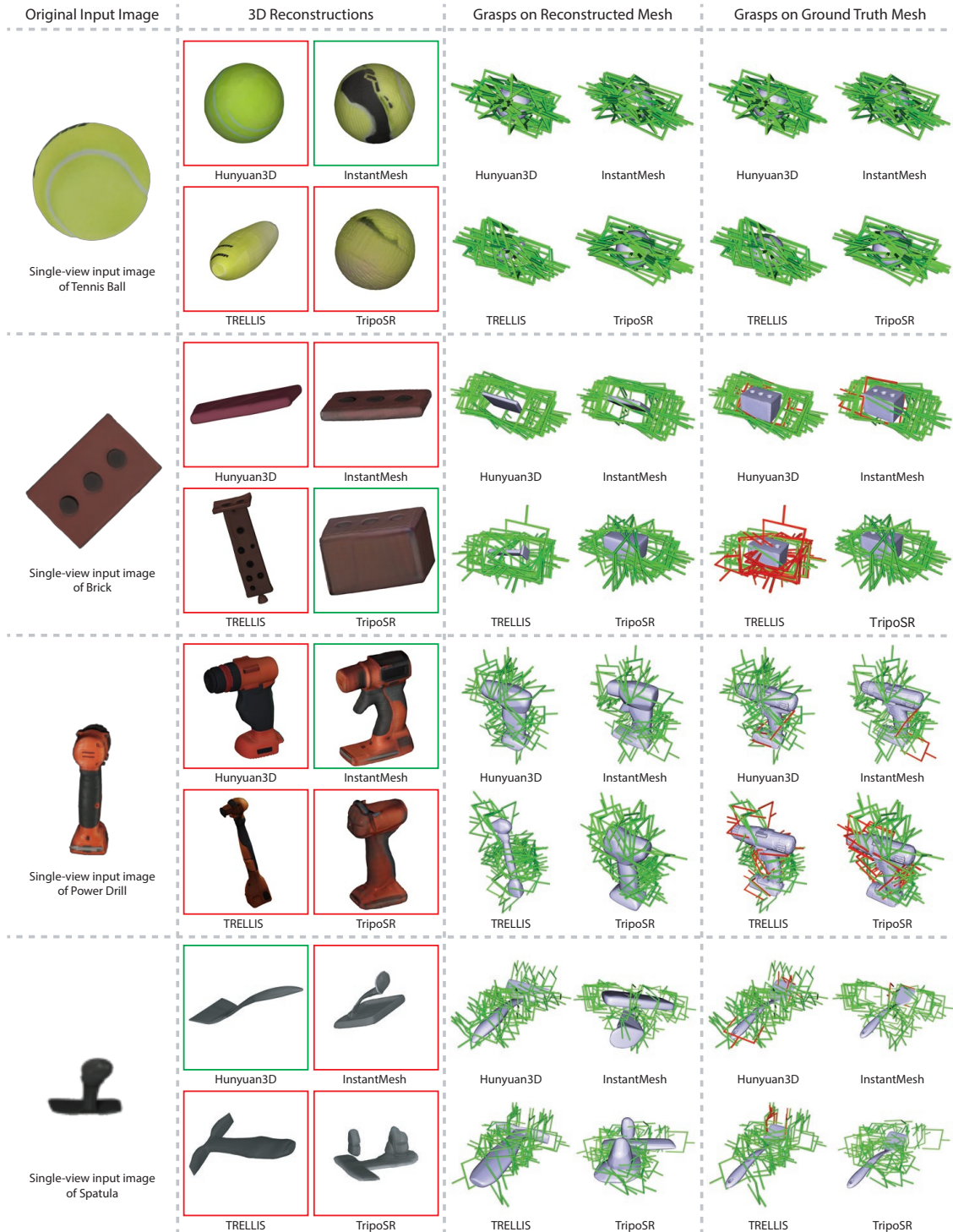


Figure 5.6: Evaluation of grasp proposals on the second half of the remaining eight YCB objects. Layout follows Figure 5.4.

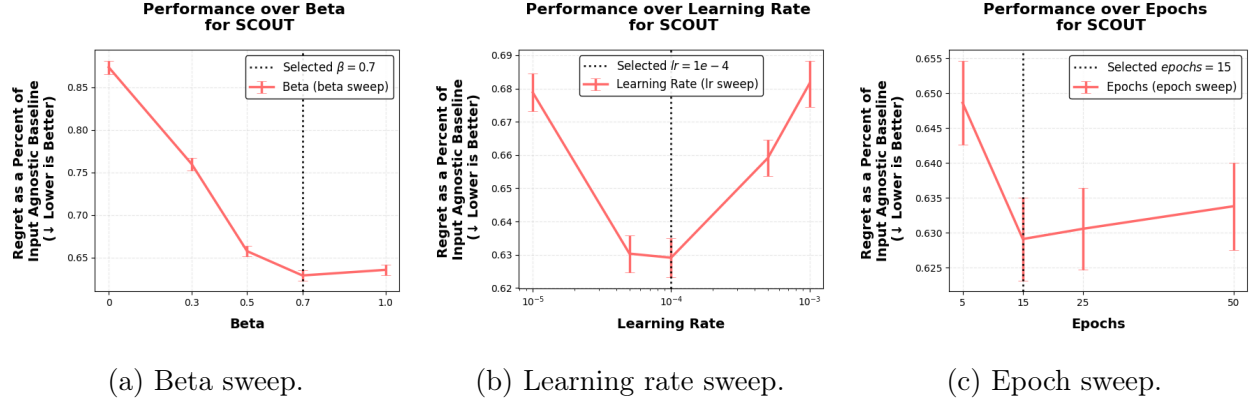


Figure 5.7: Sensitivity of SCOUT to the three hyperparameters affecting the relative-performance model $\hat{\mathbf{p}}$, evaluated on novel GSO objects under the cost function $c_{\text{view, no-cost}}$. Each curve exhibits a clear local optimum and smooth degradation away from it.

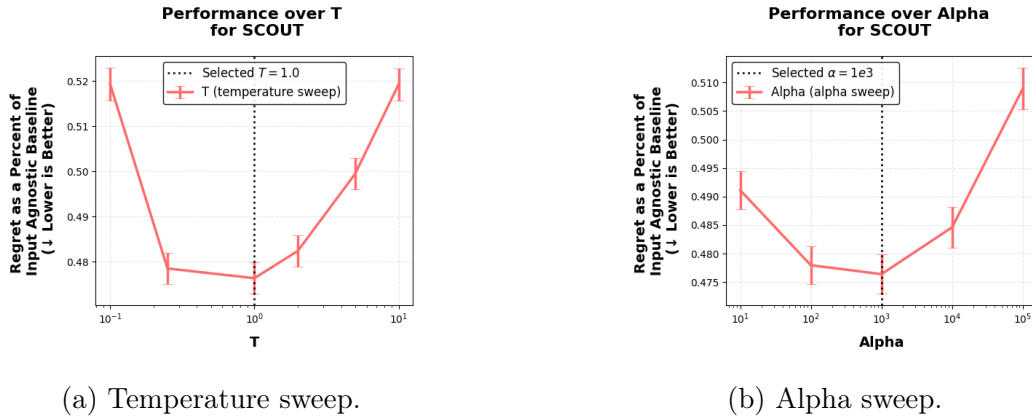


Figure 5.8: Sensitivity of SCOUT to the two hyperparameters affecting the partition function model $\hat{\mathbf{z}}$, evaluated on novel GSO objects under the cost-function family $\mathcal{C}$. Each curve exhibits a clear local optimum and smooth degradation away from it.

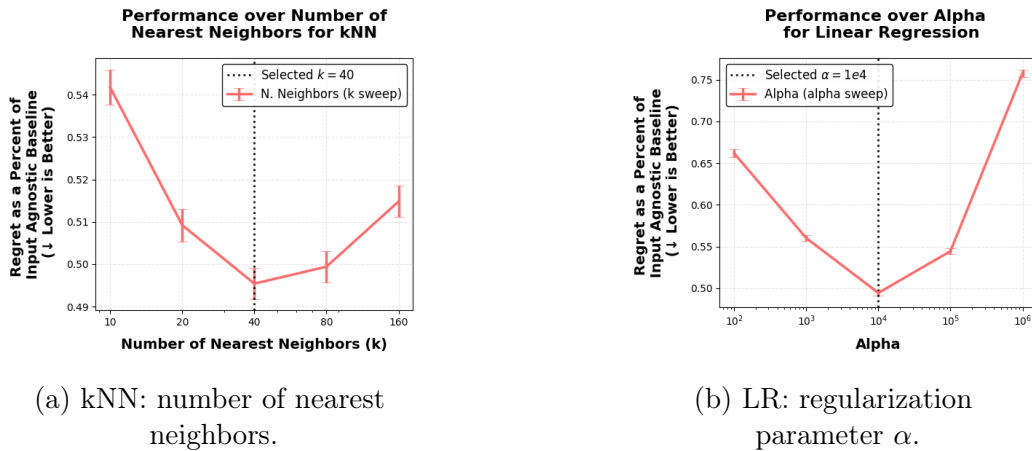
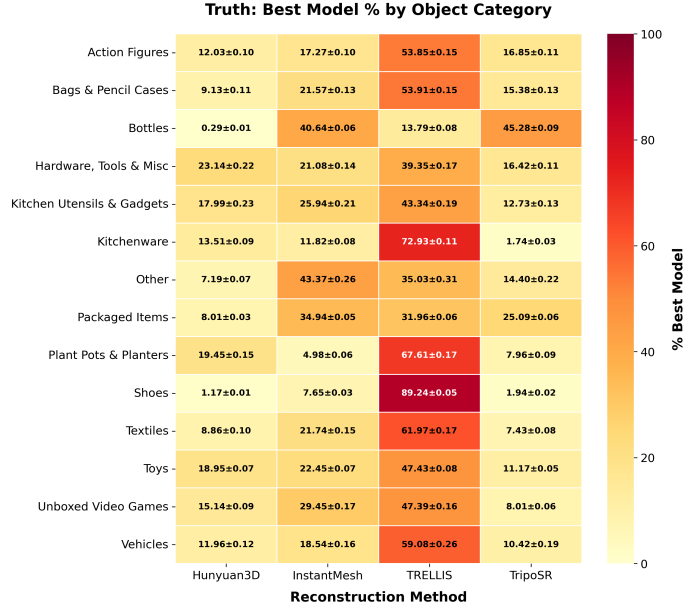
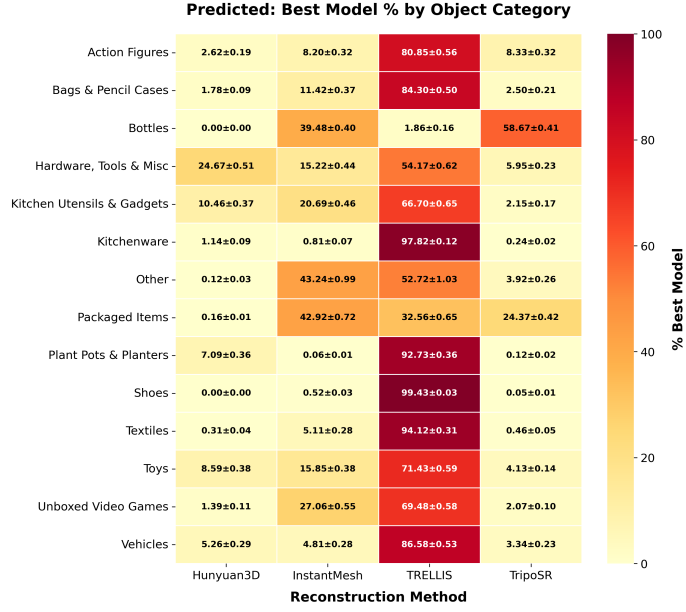


Figure 5.9: Sensitivity of the two strongest baselines to their primary hyperparameters, evaluated on novel GSO objects under the cost function family $\mathcal{C}$.

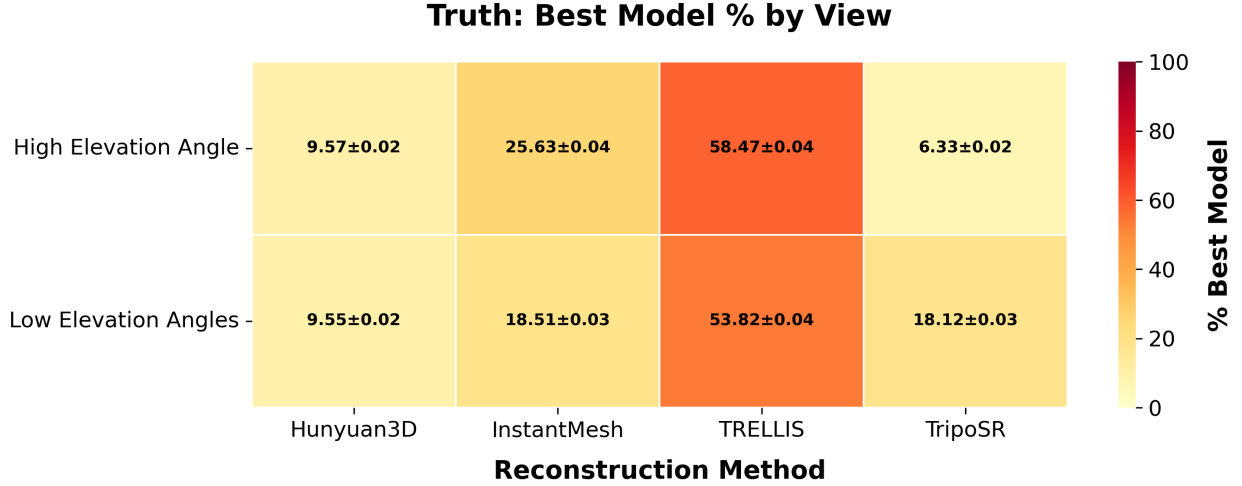


(a) Oracle policy: how often each model is truly optimal when evaluated on unseen objects, grouped by object category.

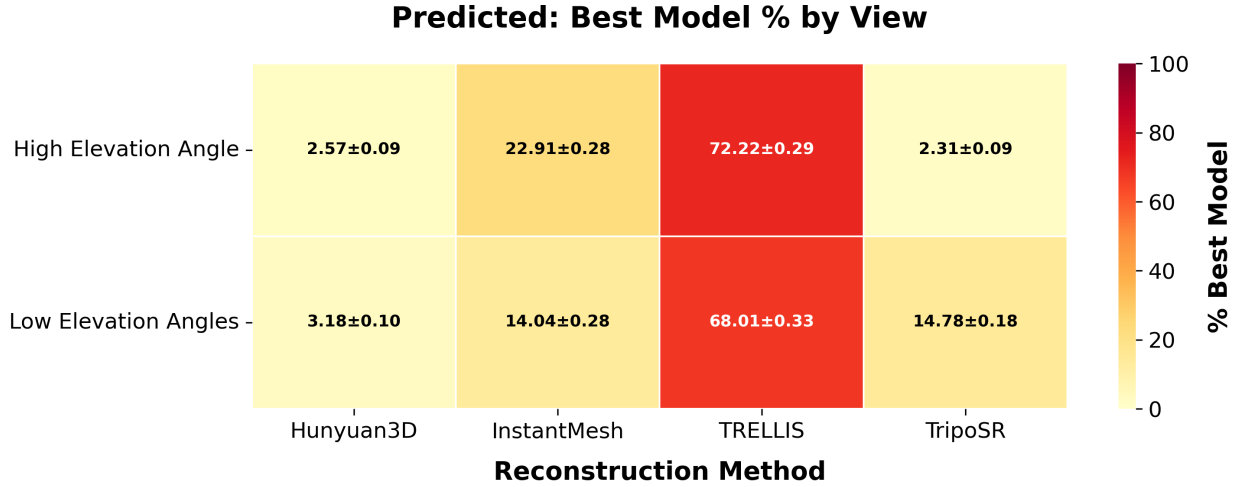


(b) SCOUT's learned policy: how often each model is selected when evaluated on unseen objects, grouped by object category.

Figure 5.10: Oracle and learned routing policies across object categories. The learned policy closely tracks the oracle's category-dependent preferences, indicating that SCOUT generalizes the category structure to novel objects rather than memorizing per-object selections.



(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 5.11: Oracle and learned routing policies across viewpoint elevations. High: 75° and 85°; low: 35°, 45°, and 60°. The learned policy reproduces the oracle's viewpoint-dependent structure, confirming that SCOUT captures viewpoint-conditional model preferences and applies them to novel objects.

Chapter 6

Conclusion

6.1 Summary

This thesis introduced SCOUT, an input-aware routing framework for selection among 3D reconstruction models on a per-input basis under user-specified cost constraints, motivated by the resource requirements of robotic manipulation pipelines. This framework is grounded in a key structural observation: as single-image 3D reconstruction has evolved into a diverse ecosystem of models, no individual model consistently dominates across the range of inputs encountered in deployment, and exhaustively querying all available models remains computationally prohibitive. In robotic manipulation settings, this challenge becomes particularly consequential, as the selected reconstruction model directly influences both robustness (how reliably the pipeline handles diverse inputs) and efficiency (whether it meets the latency and resource constraints of real-time operation).

The principal technical contribution of this work is a decoupling procedure that decomposes reconstruction scores into two interpretable components: the relative performance of viewpoint-dependent models, represented as a learned probability distribution, and the overall difficulty of the input image, represented by a scalar estimate of the partition function.

The empirical evaluation presented in Chapter 5 validates the effectiveness of the proposed framework across multiple dimensions. On Google Scanned Objects, SCOUT achieved statistically significant reductions in regret relative to adapted LLM-routing baselines across three of the four evaluated cost regimes. On BigBIRD + YCB, it attained the lowest regret on five of seven mesh quality metrics. Across the one-dimensional cost families evaluated using AIQ, SCOUT achieved the highest score in all three families considered.

Ablation studies further substantiated the contribution of each architectural component. The proposed decoupling procedure accounts for the performance gap between SCOUT and the experiments showcasing SCOUT without decoupling. Further, tag-based smoothing improved relative-performance prediction while degrading absolute-score prediction in baseline methods, thereby validating its asymmetric application within the framework. In addition, the inverse-variance weighted proxy consistently outperformed three natural alternatives on six of seven mesh reconstruction metrics.

Beyond standard benchmark evaluation, SCOUT demonstrated strong performance on downstream robotic manipulation tasks. On a 10-object grasp proposal benchmark, it

achieved the lowest mean grasp collision rate and the highest mean mesh-IoU relative to all single-model baselines. In simulation-based dexterous manipulation experiments, it attained the highest mean task success rate relative to all single-model baselines. In real-world pick-and-place experiments using a Franka Panda robot, SCOUT was able to improve the mean success rates relative to all single-model baselines. These experiments demonstrate the downstream robotic benefits that arise from routing.

Lastly, analysis of the learned routing policy indicates that SCOUT reproduces the oracle’s preferences on previously unseen objects. This suggests that the observed performance gains arise from SCOUT’s ability to capture interpretable and generalizable structure within the input distribution, rather than from per-instance memorization.

6.2 Future Work

The framework developed in this thesis admits several natural extensions. Three particularly promising directions are discussed below.

One avenue for future work includes closed-loop active perception. SCOUT performs routing at a single decision point: given an input image, it selects a reconstruction model and commits to the resulting output. A natural extension is to formulate routing as a sequential decision-making process. Upon observing a low-score prediction for all models, a robotic system could acquire additional viewpoints, re-evaluate the routing decision using the augmented input, and determine whether to commit or continue gathering information. This reframes routing as an active perception policy that explicitly trades sensing cost against expected reconstruction quality. The view-invariant methods included in SCOUT’s model pool already represent extreme points of this trade-off, incurring substantial sensing cost in exchange for high quality mesh reconstruction. However, the intermediate regime — in which one or two strategically selected additional views are acquired before re-routing — remains unexplored. Such a formulation would naturally draw upon techniques from active learning and partially observable decision processes.

Moreover, routing across the full perception pipeline enables dynamic routing selection at each stage, which would improve overall end-to-end task performance. The central motivation for reconstruction routing — that diverse models exhibit complementary strengths and no single model uniformly dominates — extends naturally to other components of modern robotic perception pipelines such as segmentation, pose estimation, and grasp proposal generation. Extending input-aware routing to these settings would introduce new technical challenges, including the need to model interactions between routing decisions across modules. For example, the optimal routing decision for reconstruction may depend critically on the preceding segmentation choice. Efficiently coordinating such interdependent decisions while sharing computation across the perception stack presents a substantial research challenge. Nonetheless, joint routing across multiple perception modules represents a promising direction with potentially significant practical benefits as the number and diversity of available robotic perception models continue to increase.

Finally, online adjustment of routing decisions during deployment would enable SCOUT to improve continuously from operational experience. SCOUT is currently trained on a fixed dataset and applies the resulting policy without modification at deployment. In practice,

robotic systems accumulate substantial information during operation, such as which models succeed on which inputs, which tasks fail, and how cost preferences should change in response. Two directions for online adjustment merit investigation. The first is online adaptation of the router itself. The routing model can be continually updated using observed deployment outcomes, effectively treating routing as a continual learning problem. The second is adaptive cost adjustment based on observed task outcomes. For example, the routing model may increase the cost budget to allow for more expensive models when downstream tasks fail, and reduce the cost budget when tasks succeed consistently.

6.3 Impact

The diversity of available 3D reconstruction models should be regarded as a source of opportunity rather than a limitation. The complementary strengths exhibited by different models create meaningful opportunities for per-input selection, enabling performance gains that cannot be achieved by any single-model pipeline. SCOUT operationalizes this insight within the context of robotic manipulation.

More broadly, this thesis points towards a future in which robotic perception pipelines are no longer conceived as monolithic stacks composed of fixed model choices, but rather as adaptive systems capable of selecting among diverse computational capabilities in response to input characteristics, task requirements, and resource constraints. Per-input model routing is a small but meaningful step in that direction, and we hope this work makes it practical and accessible for the reconstruction component of robotic manipulation pipelines.

References

- [1] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. *Zero-1-to-3: Zero-shot One Image to 3D Object*. 2023. arXiv: [2303.11328 \[cs.CV\]](#).
- [2] R. Shi, H. Chen, Z. Zhang, M. Liu, C. Xu, X. Wei, L. Chen, C. Zeng, and H. Su. “Zero123++: a single image to consistent multi-view diffusion base model.” *arXiv preprint arXiv:2310.15110*, 2023.
- [3] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. “DreamFusion: Text-to-3D using 2D Diffusion.” *arXiv*, 2022.
- [4] H. Wang, X. Du, J. Li, R. A. Yeh, and G. 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*. 2023, pp. 12619–12629.
- [5] J. Seo, W. Jang, M.-S. Kwak, H. Kim, J. Ko, J. Kim, J.-H. Kim, J. Lee, and S. Kim. “Let 2d diffusion model know 3d-consistency for robust text-to-3d generation.” *arXiv preprint arXiv:2303.07937*, 2023.
- [6] G. Qian, J. Mai, A. Hamdi, J. Ren, A. Siarohin, B. Li, H.-Y. Lee, I. Skorokhodov, P. Wonka, S. 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.
- [7] 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. arXiv: [2403.02151 \[cs.CV\]](#). URL: <https://arxiv.org/abs/2403.02151>.
- [8] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. “Lrm: Large reconstruction model for single image to 3d.” *arXiv preprint arXiv:2311.04400*, 2023.
- [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. arXiv: [2404.07191 \[cs.CV\]](#). URL: <https://arxiv.org/abs/2404.07191>.
- [10] M. Liu, C. Xu, H. Jin, L. Chen, M. Varma T, Z. Xu, and H. 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**, 2023, pp. 22226–22246.
- [11] 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.

- [12] T. H. Team. *Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation*. 2025. arXiv: [2501.12202 \[cs.CV\]](#).
- [13] F. Nolte, A. Geiger, B. Schölkopf, and I. Posner. “Is single-view mesh reconstruction ready for robotics?” *arXiv preprint arXiv:2505.17966*, 2025.
- [14] B. Sen, A. Agarwal, G. Singh, S. Sridhar, M. Krishna, et al. “Scarp: 3d shape completion in arbitrary poses for improved grasping.” *arXiv preprint arXiv:2301.07213*, 2023.
- [15] J. Lundell, F. Verdoja, and V. Kyrki. “Beyond top-grasps through scene completion.” In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 545–551.
- [16] S. S. Mohammadi, N. F. Duarte, D. Dimou, Y. Wang, M. Taiana, P. Morerio, A. Dehban, P. Moreno, A. Bernardino, A. Del Bue, et al. “3dsgrasp: 3d shape-completion for robotic grasp.” *arXiv preprint arXiv:2301.00866*, 2023.
- [17] A. Agarwal, G. Singh, B. Sen, T. Lozano-Pérez, and L. P. Kaelbling. “SceneComplete: Open-World 3D Scene Completion in Cluttered Real World Environments for Robot Manipulation.” *IEEE Robotics and Automation Letters*, **11**(1), 2026, pp. 482–489. DOI: [10.1109/LRA.2025.3630884](#).
- [18] A. Kashyap, Y. Yang, H. Andreasson, and T. Stoyanov. “Single-View Shape Completion for Robotic Grasping in Clutter.” *arXiv preprint arXiv:2512.16449*, 2025.
- [19] S. Iwase, M. Z. Irshad, K. Liu, V. Guizilini, R. Lee, T. Ikeda, A. Amma, K. Nishiwaki, K. Kitani, R. Ambrus, et al. “ZeroGrasp: Zero-shot shape reconstruction enabled robotic grasping.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2025, pp. 17405–17415.
- [20] S. Iwase, K. Liu, V. Guizilini, A. Gaidon, K. Kitani, R. Ambrus, and S. Zakharov. “Zero-shot multi-object scene completion.” In: *European Conference on Computer Vision*. Springer. 2024, pp. 96–113.
- [21] 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*. 2023, pp. 17739–17749.
- [22] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu. “Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction.” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 3295–3306.
- [23] 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*. 2024, pp. 1–11.
- [24] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola. “Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation.” In: *7th Annual Conference on Robot Learning*. 2023.
- [25] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg. “Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects.” In: *6th Annual Conference on Robot Learning*. 2022.

- [26] K. Lu, H. Yuan, R. Lin, J. Lin, Z. Yuan, C. Zhou, and J. 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)*. 2024, pp. 1964–1974.
- [27] S. Chen, W. Jiang, B. Lin, J. Kwok, and Y. Zhang. “Routerdc: Query-based router by dual contrastive learning for assembling large language models.” *Advances in Neural Information Processing Systems*, **37**, 2024, pp. 66305–66328.
- [28] 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.
- [29] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. “Routellm: Learning to route llms with preference data.” *arXiv preprint arXiv:2406.18665*, 2024.
- [30] Y. Li. “Rethinking Predictive Modeling for LLM Routing: When Simple kNN Beats Complex Learned Routers.” *arXiv preprint arXiv:2505.12601*, 2025.
- [31] 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)*. Ieee. 2022, pp. 2553–2560.
- [32] 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)*. IEEE. 2014, pp. 509–516.
- [33] 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)*. IEEE. 2015, pp. 510–517.
- [34] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin. “Density-aware chamfer distance as a comprehensive metric for point cloud completion.” *arXiv preprint arXiv:2111.12702*, 2021.
- [35] S. Duggal, Y. Hu, O. Michel, A. Kembhavi, W. T. Freeman, N. A. Smith, R. Krishna, A. Torralba, A. Farhadi, and W.-C. Ma. “Eval3D: Interpretable and fine-grained evaluation for 3D generation.” In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 13326–13336.
- [36] M. Liu, R. Shi, L. Chen, Z. Zhang, C. Xu, X. Wei, H. Chen, C. Zeng, J. Gu, and H. 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*. 2024, pp. 10072–10083.
- [37] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “Nerf: Representing scenes as neural radiance fields for view synthesis.” *Communications of the ACM*, **65**(1), 2021, pp. 99–106.

- [38] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. “Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction.” *arXiv preprint arXiv:2106.10689*, 2021.
- [39] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, et al. “3d gaussian splatting for real-time radiance field rendering.” *ACM Trans. Graph.*, **42**(4), 2023, pp. 139–1.
- [40] X. Tu and K. Desingh. “SuperQ-GRASP: Superquadrics-based Grasp Pose Estimation on Larger Objects for Mobile-Manipulation.” In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 3361–3368.
- [41] L. Zhou, H. Wang, Z. Zhang, Z. Liu, F. E. Tay, and adn Marcelo H. Ang. Jr. *You Only Scan Once: A Dynamic Scene Reconstruction Pipeline for 6-DoF Robotic Grasping of Novel Objects*. 2024. arXiv: [2404.03462](https://arxiv.org/abs/2404.03462) [cs.CV]. URL: <https://arxiv.org/abs/2404.03462>.
- [42] Y. Mu et al. “RoboTwin: Dual-Arm Robot Benchmark with Generative Digital Twins.” In: *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*. June 2025, pp. 27649–27660.
- [43] K. Wang et al. *ManiTwin: Scaling Data-Generation-Ready Digital Object Dataset to 100K*. 2026. arXiv: [2603.16866](https://arxiv.org/abs/2603.16866) [cs.R0]. URL: <https://arxiv.org/abs/2603.16866>.
- [44] G. Pan and H. Wang. “A Cost-Benefit Analysis of On-Premise Large Language Model Deployment: Breaking Even with Commercial LLM Services.” *arXiv preprint arXiv:2509.18101*, 2025.
- [45] D. W. Shu, S. W. Park, and J. Kwon. “3d point cloud generative adversarial network based on tree structured graph convolutions.” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3859–3868.
- [46] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. “Learning Representations and Generative Models for 3D Point Clouds.” In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 40–49. URL: <https://proceedings.mlr.press/v80/achlioptas18a.html>.
- [47] 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*. PmLR. 2021, pp. 8748–8763.
- [48] 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*. 2016, pp. 770–778.
- [49] 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.
- [50] 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*. 2022, pp. 11976–11986.

- [51] Tencent Hunyuan Team. *Hunyuan3D-2 source code repository*. <https://github.com/Tencent-Hunyuan/Hunyuan3D-2>. 2025.
- [52] 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.
- [53] 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.
- [54] 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.
- [55] 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)*, **41**(4), 2022, pp. 1–18.
- [56] A. Devert. *miniball: Efficient computation of the smallest bounding ball of a point set*. <https://github.com/marmakoide/miniball>. Python implementation of Welzl’s algorithm. 2023.
- [57] 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*. 2024, pp. 17868–17879.
- [58] 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.
- [59] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. “Learning ambidextrous robot grasping policies.” *Science Robotics*, **4**(26), 2019, eaau4984. DOI: [10.1126/scirobotics.aau4984](https://doi.org/10.1126/scirobotics.aau4984). eprint: <https://www.science.org/doi/pdf/10.1126/scirobotics.aau4984>. URL: <https://www.science.org/doi/abs/10.1126/scirobotics.aau4984>.