

Supplementary - GRAFT: Graph-Based Affordance Transfer via Part Correspondence

Mengying Lin¹ Utkarsh Mishra¹ Ajay Mandlekar² Danfei Xu¹

¹Georgia Institute of Technology ²NVIDIA

{mlin365, umishra31, danfei}@gatech.edu

{amandlekar}@nvidia.com

1. Implementation Details

1.1. Assets used for evaluation

In our zero-shot manipulation generalization experiments, we use the object assets listed in Table 1. Because the PartNet-Mobility dataset provides only coarse category labels and objects within the same category often exhibit substantial geometric and functional variation, we refine these labels for our evaluation. In Table 1, the *Raw Category* column shows the original dataset labels, while the *Relabeled Category* column reflects the refined categories used in our experiments.

Split	Raw Category	ID	Relabeled Category
Seen Objects			
Seen	Bucket	100454	WaterBucket
Seen	Kettle	102715	Teapot
Seen	Bottle	3558	WineBottle
Seen	Bottle	8848	Mug
Seen	Scissors	100144	Pliers
Unseen Objects			
Unseen	Bucket	100466	ShoppingBasket
Unseen	Dispenser	103357	Detergent
Unseen	Dispenser	101560	Shampoo
Unseen	Dispenser	101417	Spray
Unseen	Kettle	102765	Kettle
Unseen	Suitcase	100249	Briefcase
Unseen	Suitcase	101681	Handbag
Unseen	Phone	103251	Phone
Unseen	Remote	104040	Remote
Unseen	Scissors	10537	Scissors
Unseen	Scissors	11026	GardenShears

Table 1. **Object Instances Used in Zero Shot Affordance Generalization Experiments.** We list all seen objects (used for demonstration collection) and unseen objects (used for zero-shot evaluation), along with their PartNet-Mobility categories and IDs.

For the physics-based simulation evaluation, we further narrow the object set based on practical feasibility. Specifically, whether the Franka gripper can reliably grasp the object and whether the mesh satisfies requirements such as being watertight and free of severe artifacts. The subset used for simulation experiments is provided in Table 2.

Split	Raw Category	ID	Relabeled Category
Seen Objects			
Seen	Bucket	100454	WaterBucket
Seen	Kettle	102715	Teapot
Seen	Bottle	3558	WineBottle
Seen	Bottle	8848	Mug
Unseen Objects			
Unseen	Dispenser	103357	Detergent
Unseen	Dispenser	101560	Shampoo
Unseen	Dispenser	101417	Spray
Unseen	Kettle	102765	Kettle
Unseen	Suitcase	100249	Briefcase
Unseen	Suitcase	101681	Handbag
Unseen	Phone	103251	Phone
Unseen	Remote	104040	Remote

Table 2. **Object Instances Used in Physics-based Simulation.**

1.2. Hyperparameters

We summarize the hyperparameters used in our algorithm.

UFGW formulation. We use a feature–structure tradeoff of $\alpha = 0.5$, entropic regularization $\varepsilon = 0.02$, and unbalanced transport penalties $\lambda_s = \lambda_t = 0.5$. The unmatched-mass (KL) penalty is weighted by $w_{\text{unmatched}} = 0.8$. The unary cost combines semantic and bounding box extent distance terms: $w_{\text{sem}} = 0.4$, $w_{\text{extent}} = 0.3$, and structural (edge-based) cost uses weight $w_{\text{edge}} = 0.3$.

Graph normalization and descriptors. Each object graph is normalized by its bounding-box scale. Nodes store normalized OBB center, extent, and axis; L2-normalized DINO features; and corresponding part point cloud.

BFS-based saliency. Functional saliency is propagated from annotated interactable parts using a discounted BFS with decay factor $\gamma_{\text{BFS}} = 0.6$.

Target saliency seeding. For the target graph, saliency is initialized from unary similarity to the source’s interactable part using

$$s_B(j) \propto \exp \left[-\beta (c_{ij} - \min_k c_{ik}) \right], \quad \beta = 8.0,$$

followed by normalization.

EM-style soft anchoring. We run 5 EM iterations to refine the posterior over the target root node. Each update mixes posterior and prior with coefficient $\tau = 0.4$. After convergence, the best root node is anchored and we solve a reduced UFGW problem.

Match extraction. Discrete node correspondences are extracted from the final transport plan using a threshold ratio of 0.5 relative to row/column mass.

1.3. Parsing the Real-world Object Mesh

Mesh Scanning. We collect 600 frames using a free Play-Store App for 3D Mesh Generation [5]. After processing, the application provides us with an object-centric OBJ file.

Part annotation. To enable part level correspondence with our simulation objects, we manually annotate the mesh using Blender. Following the same conventions as in simulation, we segment the object into functional components (for example, body, handle, spout) using geometric boundaries and curvature cues. The annotation is saved as per part vertex groups, from which we derive per part oriented bounding boxes and part point clouds.

Correspondence based grasp transfer. With the annotated real world mesh, we query our simulation demonstration set to identify the most compatible source object using GRAFT. We compute part level correspondence between the real mesh and each source graph and select the one with minimum UFGW distance. The matched source grasping point is then transferred to the real object via the part level and vertex level correspondence chain. This yields an estimated grasping point p^{target} defined in the canonical frame of the scanned mesh.

Pose estimation and grasp prediction. To execute the grasp on a physical robot, we estimate the six degree of freedom pose of the real object in the scene using FoundationPose [7]. FoundationPose supports multi view RGBD fusion, so we directly feed RGBD images captured from the Azure Kinect camera without requiring the mesh as an input model. The predicted object pose ${}^{\text{cam}}T_{\text{obj}}$ allows us to transform the transferred grasping point into the camera frame and subsequently into the Franka Panda base frame.

Given the transformed grasp location, we employ Any-Grasp [1] to compute a physically feasible grasp around the transferred point. AnyGrasp takes object point cloud, and returns a set of candidate gripper poses. We first filter these poses using a predefined distance threshold to the predicted target point \hat{p} , and select the highest-confidence remaining pose and send it to the robot controller for execution.

Robot execution. The final grasp is executed on a Franka Panda arm. The robot follows a two stage motion: (i) an approach trajectory that moves the end effector to a pre grasp pose offset by 5cm along the gripper axis, and (ii) a straight line insertion toward the grasping point, followed by finger closure. If the grasp is stable, the robot lifts the object by 15 cm to validate success.

2. Definition of Metrics for Evaluating Zero Shot Affordance Generalization

We extend the 2D affordance metrics from ROBO-ABC [4] to 3D point clouds. Each test object is represented by ground-truth points $\{p_i, w_i\}$, where $p_i \in \mathbb{R}^3$ and $w_i \in [0, 1]$ denote the position and graspability weight derived from the Gaussian-smoothed annotation. Given predicted grasping indices \mathcal{P} , we compute three metrics:

Affordance Success Rate (ASR). The proportion of predicted points lying in graspable regions:

$$\text{ASR} = \frac{1}{|\mathcal{P}|} \sum_{p_i \in \mathcal{P}} \mathbb{I}[w_i > \tau],$$

where $\tau = 0.5$; points within ε distance of positive regions are also counted as successful.

Normalized Scanpath Saliency (NSS). Measures how salient predicted points are within the normalized affordance field:

$$\text{NSS} = \frac{1}{|\mathcal{P}|} \sum_{p_i \in \mathcal{P}} \frac{w_i - \mu_w}{\sigma_w},$$

where μ_w and σ_w are the mean and standard deviation of $\{w_i\}$.

Distance-To-Mass (DTM). Quantifies how far each prediction lies from the nearest graspable region:

$$\text{DTM} = \frac{1}{|\mathcal{P}|} \sum_{p_i \in \mathcal{P}} \frac{\min_{w_j > \tau} \|p_i - p_j\|_2}{d_{\text{bbox}}},$$

where distances are computed in 3D using a KD-tree over positive ground-truth points, and d_{bbox} normalizes by object scale. Predictions inside the graspable region yield DTM = 0.

3. Graspable Region Annotation Process

To construct point-level ground-truth affordance labels for our zero-shot evaluation, we manually annotate graspable regions for every test object. This section provides additional details on the annotation pipeline, the conversion to continuous affordance fields, and the preprocessing used to map annotations onto 3D point clouds.

3.1. Mesh Preparation and Import

We begin by exporting each target object mesh from the PartNet-Mobility dataset and importing it into Blender. Meshes are inspected for watertightness and cleaned when necessary (removing duplicate vertices, merging disconnected shells).

3.2. Manual Mask Annotation

Annotators paint a *binary graspable mask* directly on the mesh surface in Blender following two guidelines:

- **Only mark physically realizable grasp regions.** Flat or slightly curved surfaces that permit frictional contact are included; sharp edges, excessively thin structures, or decorative parts are excluded.
- **Mark only functionally relevant parts given a specific task.** For example, given a kettle in pick-and-place task, the handle is labeled but the spout is not.

Each mask is verified from multiple viewpoints to ensure complete surface coverage on the intended region.

3.3. Mapping to 3D Point Cloud

To evaluate affordance metrics on normalized point clouds, we project the vertex-color mask onto the point cloud sampled from the mesh:

1. Sample $\sim 5\text{k}-10\text{k}$ points uniformly from the mesh surface.
2. For each point, query the nearest triangle and interpolate the annotated vertex colors.
3. Assign a binary graspability value $m_i \in \{0, 1\}$ to each point p_i .

This produces a dense, binary graspability map on the 3D point cloud.

3.4. Continuous Affordance Field

Binary labels do not reflect the smooth spatial transition near graspable boundaries. A point slightly outside the annotated region may still be graspable depending on gripper tolerance and local surface geometry. Following the affordance and saliency literature [2, 6], where Gaussian kernels

are applied to 2D grasping points to obtain a continuous affordance field, we extend this practice to 3D. Specifically, we convert the binary mask into a continuous affordance function via Gaussian smoothing.:

$$w_i = \sum_j m_j \exp\left(-\frac{\|p_i - p_j\|_2^2}{2\sigma^2}\right),$$

where σ is set adaptively to $0.02 d_{\text{bbox}}$ (2% of the object’s bounding-box diagonal). The resulting continuous field $w_i \in [0, 1]$ serves as the ground-truth affordance score used for computing ASR, NSS, and DTM.

We provide some visual examples of annotated affordance field in Figure 1.

3.5. Normalization and Coordinate Alignment

Before metric computation, point clouds are normalized to the object canonical coordinate frame:

1. Centering the cloud using the mesh centroid.
2. Scaling coordinates by the bounding-box diagonal d_{bbox} . This ensures consistency across objects of different scales and is required for distance-based metrics such as DTM.

4. Top-K Retrieval Qualitative Results

To further illustrate the behavior of our retrieval module, we provide qualitative Top-2 retrieval results in Figure 2. Each row shows a query object (left) and the two highest-ranked source objects retrieved by three methods: (a) GRAFT (ours), (b) ROBO-ABC (DINOv2), and (c) ROBO-ABC (CLIP).

Our visual comparisons highlight three consistent trends:

First, GRAFT consistently retrieves source objects that share functional part structure with the query object. For example, the shopping basket with handle retrieves handled objects like suitcase and bucket even when they differ in overall appearance. This is because GRAFT matches objects at the part level, allowing it to recognize that handles share similar functionality.

Second, retrieval methods that rely solely on global object features often underperform when the task relevant part features are weak or absent in the holistic embedding. Global descriptors such as those from DINO or CLIP primarily capture cues like overall silhouette, large planar surfaces, strong contours, coarse object semantics, and sometimes dominant textures. As a result, they may retrieve objects that appear similar in dominant features at the whole object level but lack functional compatibility. For example, given a pick-and-place task, in ROBO ABC with DINO or CLIP retrieval, a suitcase query may return a laptop or a box simply because they share similar flat surface geometry. Such mismatched objects cannot support correct contact point transfer and therefore lead to poor downstream execution success.



Figure 1. Examples of annotated affordance field.



Figure 2. **Top-2 retrieval results visualization for lifting task.** For each query object (left), we show the two most compatible source objects retrieved by each method. GRAFT consistently retrieves functionality compatible objects whose graspable regions align with the intended manipulation.

Third, the part level reasoning in GRAFT encourages broader retrieval diversity. Because GRAFT focuses on structural similarity with functionality awareness, it retrieves objects from different semantic groups whenever they share a relevant configuration of parts. This improves both zero shot manipulation transfer and data generation diversity in later stages.

5. Compute and Wall-clock time Analysis

To quantify the computational efficiency of our correspondence pipeline, we measure the wall-clock time of its two major components: (i) graph construction, which builds the part-level object graph and extracts descriptors, and (ii) graph matching, which computes the UFGW-based correspondence between a source and target object. All experiments are conducted on an NVIDIA RTX 3090 GPU and

an Intel i9-13900K CPU, using a single thread for timing consistency.

Setup. For each component, we run the evaluation 50 independent times and report the mean, standard deviation, and observed minimum/maximum runtime. Graph construction is measured per individual object. Graph matching is measured per source–target pair.

Results. Graph construction accounts for most of the compute time. GPU monitoring indicates sparse utilization during geometry preprocessing, followed by a short plateau of sustained 100% GPU usage during DINO feature extraction. Peak memory usage remains modest at around 5.6 GB, suggesting that descriptor extraction is computationally but not memory intensive.

tag: gpu/mem_GB

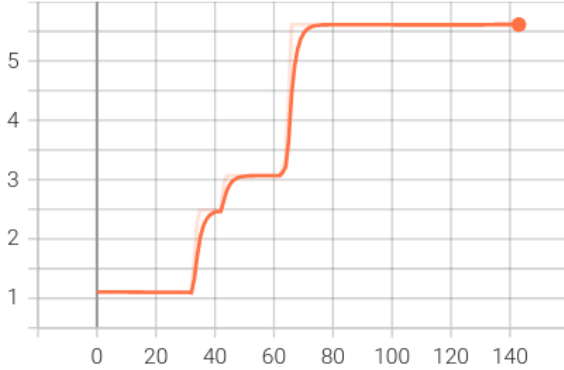


Figure 3. GPU memory usage during graph construction. Memory consumption has a peak of about 5.6 GB during DINO feature extraction. The overall memory footprint is modest.

In contrast, graph matching is extremely lightweight. The UFGW solver converges quickly, requiring only a few Sinkhorn iterations and averaging approximately 2 ms per source–target pair. This efficiency allows retrieval to scale gracefully to large demonstration repositories, keeping total retrieval time well below one second even when comparing against hundreds of stored objects.

Overall, these measurements show that the offline graph construction step is the main computational bottleneck, whereas online correspondence and retrieval are effectively real-time. This separation ensures that GRAFT remains practical for deployment scenarios where fast test-time transfer is required.

Component	Mean (s)	Std (s)	Min (s)	Max (s)
Graph Construction	11.31	1.60	9.49	14.56
Graph Matching	0.00207	0.00066	0.00100	0.00376

Table 3. Wall-clock performance of the major components in our correspondence pipeline. Graph construction dominates the compute time due to part extraction and descriptor computation, taking around 11.3 seconds per object. The graph matching algorithm runs in approximately 2 milliseconds per source-target pair.

6. Robustness Analysis

6.1. Segmentation Assumption and Robustness

Our method assumes access to part-level segmentation, which is orthogonal to the contribution of correspondence learning. To evaluate a fully automated pipeline, we integrate GRAFT with SUPERDEC [3], a 3D superquadric decomposition method.

As shown in Table 4, GRAFT combined with automatic segmentation already outperforms the strongest baseline,

tag: gpu/util

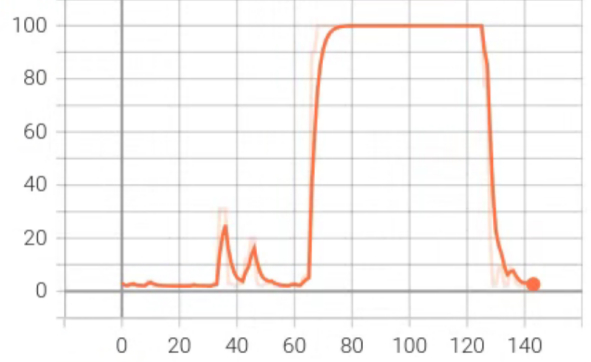


Figure 4. GPU utilization over time when constructing a part-graph. Utilization stays low during pre-processing, then spikes to sustained 100% during forward passes of the vision foundation model.

while ground-truth segmentation provides an upper bound. Although over- or under-segmentation can affect performance, these results suggest that our method is robust to realistic segmentation noise and can benefit from future advances in part decomposition.

Methods	ASR \uparrow	NSS \uparrow	DTM \downarrow
ROBO-ABC [4] (DINO+DIFT)	0.69	0.48	0.10
GRAFT+ SUPERDEC [3]	0.73	1.20	0.06
GRAFT+ GT	0.85	0.82	0.02

Table 4. Affordance generalization results with ground truth part annotations and annotations generated by SUPERDEC.

6.2. Hyperparameter Robustness

We evaluate robustness by applying Gaussian perturbations to key hyperparameters, including the structure-feature tradeoff and matching cost weights.

As shown in Table 5, performance degrades gracefully under increasing noise levels, indicating that the method is not overly sensitive to precise parameter tuning.

Additionally, ablations in the main paper demonstrate that removing key components leads to clear performance drops, confirming their importance.

Noise Scale	ASR (%) \uparrow	NSS \uparrow	DTM \downarrow
0.02	83.64 \pm 3.64	0.82 \pm 0.05	0.02 \pm 0.01
0.05	85.45 \pm 4.45	0.82 \pm 0.09	0.02 \pm 0.01
0.10	80.91 \pm 4.90	0.81 \pm 0.13	0.03 \pm 0.01
0.20	77.27 \pm 6.10	0.75 \pm 0.18	0.04 \pm 0.02

Table 5. Sensitivity analysis under Gaussian perturbations applied to key hyperparameters (structure-feature tradeoff and cost weights). Results are averaged over 10 runs.

7. Extension to Articulated Objects

Although our main experiments focus on rigid objects, the proposed framework is not inherently restricted to them.

We demonstrate this by transferring a drawer-opening trajectory to an air-fryer with a similar articulated structure. The model successfully identifies corresponding functional parts and transfers the interaction trajectory.

This example highlights the potential of our method to generalize beyond rigid objects. Future work will include systematic evaluation on articulated manipulation tasks.

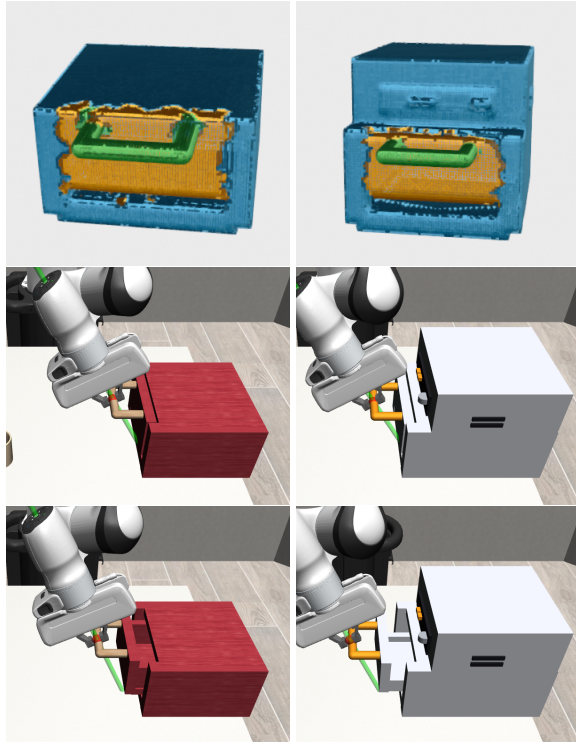


Figure 5. Extension to articulated objects.

References

- [1] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39(5):3929–3945, 2023. [2](#)
- [2] Kuan Fang, Te-Lin Wu, Daniel Yang, Silvio Savarese, and Joseph J Lim. Demo2vec: Reasoning object affordances from online videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2139–2147, 2018. [3](#)
- [3] Elisabetta Fedele, Boyang Sun, Leonidas Guibas, Marc Pollefeys, and Francis Engelmann. Superdec: 3d scene decomposition with superquadric primitives. *arXiv preprint arXiv:2504.00992*, 2025. [5](#)
- [4] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. In *European Conference on Computer Vision*, pages 222–239. Springer, 2024. [2](#), [5](#)
- [5] Laan Labs. 3d scanner app – lidar scanner for ipad and iphone pro. <https://3dscannerapp.com/>. Accessed: 2025-11-14. [2](#)
- [6] Hongchen Luo, Wei Zhai, Jing Zhang, Yang Cao, and Dacheng Tao. Learning affordance grounding from exocentric images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2252–2261, 2022. [3](#)
- [7] Evin Pinar Örnek, Yann Labbé, Bugra Tekin, Lingni Ma, Cem Keskin, Christian Forster, and Tomas Hodan. Foundpose: Unseen object pose estimation with foundation features. In *European Conference on Computer Vision*, pages 163–182. Springer, 2024. [2](#)