

Supplemental Material

A. Framework Details

A.1. Affordance Flow Matching

The training pipeline of Affordance Flow Matching (AFM) is shown in Figure 1 (a). A noise Ω_0 is first sampled from a Gaussian distribution with zero mean and one unit. A timestep t is sampled from 0 to unit variance. Then, the training input Ω_t is obtained by $\Omega_t = (1 - t) * \Omega_0 + t * \Omega_1$, where Ω_1 is the ground truth of the affordance. The scene point clouds \mathcal{O} are fed into PointNet++ [13] with several upsampling and downsampling layers to obtain features $\mathcal{F}_{obj} \in \mathbb{R}^{N_{scene} \times C_{scene}}$. The object features are then concatenated with Ω_t and point cloud positions \mathcal{O} , and processed by an MLP layer to obtain affordance features $\mathcal{F}_{aff} \in \mathbb{R}^{N_{scene} \times C_{aff}}$. The direction vector, a one-hot vector of 6 dimensions, is fed into an MLP to obtain the direction features $\mathcal{F}_{dir} \in \mathbb{R}^{1 \times C_{dir}}$. The language text is encoded by the CLIP model [14] and outputs language features $\mathcal{F}_{lan} \in \mathbb{R}^{1 \times C_{lan}}$. These features are then fed into Perceiver IO [8, 9] for effective feature interaction. Specifically, the direction and language features serve as the query, while the affordance features serve as the key and value for cross-attention. Then, the output of the first attention mechanism performs self-attention. Then, the scene features serve as the query and the self-attention result serve as key and value in the final cross-attention to obtain the final features. The features are then fed into an MLP to predict the flow $v_\eta(\Omega_t, t|c_1)$. The loss function is the L2 loss between predicted flow with the target:

$$\mathcal{L}_{AFM} = \|v_\eta(\Omega_t, t|c) - (\Omega_1 - \Omega_0)\|^2. \quad (1)$$

A.2. Grasp Flow Matching

The training pipeline of Grasp Flow Matching (GFM) is shown in Figure 1. The object and affordance are concatenated first then fed into PointNet++ with several downsampling layer to obtain the affordance features. The direction and language are encoded same with AFM. And all features are concatenated in the token dimension to obtain the condition features. The noised grasp poses are obtained similar with AFM, which are encoded by MLP. Then the pose features serve as query, and the condition features serve as key and value in the transformer decoder. The output features are fed into an MLP to obtain the prediction pose flow. Then the grasp pose can be obtained by one step, during the training time:

$$\mathcal{G}_1^{dex} = \mathcal{G}_t^{dex} + (1 - t) * v_\theta(\mathcal{G}_t^{dex}, t|c_2), \quad t \in [0, 1), \quad (2)$$

The loss functions includes grasp pose regression loss, hand chamfer loss and hand fingertip key point loss, which are

detailed in next section.

A.3. Loss Function

This section provides a detailed exposition of the loss functions utilized during the grasp generation and optimization.

- Parameter Regression Loss. We utilize the mean squared error (MSE) to quantify the deviation between the generated dexterous hand pose \mathcal{G}_1^{dex} and the ground truth \mathcal{G}_1^{dex} .

$$\mathcal{L}_{pose} = \|\mathcal{G}_1^{dex} - \hat{\mathcal{G}}_1^{dex}\|_2^2. \quad (3)$$

- Hand Chamfer Loss. The predicted hand point clouds $\mathcal{H}(\hat{\mathcal{G}}_1^{dex})$ and the ground truth $\mathcal{H}(\mathcal{G}_1^{dex})$ are derived by sampling from the hand mesh. We then compute the chamfer distance to assess the discrepancies between the predicted and ground-truth hand shapes.

$$\begin{aligned} \mathcal{L}_{chamfer} = & \sum_{x \in \mathcal{H}(\mathcal{G}^{dex})} \min_{y \in \mathcal{H}(\hat{\mathcal{G}}_1^{dex})} \|x - y\|_2^2 \\ & + \sum_{x \in \mathcal{H}(\hat{\mathcal{G}}^{dex})} \min_{y \in \mathcal{H}(\mathcal{G}_1^{dex})} \|x - y\|_2^2. \end{aligned} \quad (4)$$

- Fingertip Keypoint Loss. The fingertip keypoint loss \mathcal{L}_{tip} measures the distance between the fingertip of generated grasp q^g and the ground truth q^{gt} .

$$\mathcal{L}_{tip} = \|q^g - q^{gt}\|_2^2. \quad (5)$$

- Affordance contact loss. The affordance contact loss is designed for constraining hand contact candidate points in contact with the affordance area:

$$\mathcal{L}_{aff-dist} = \sum_i \mathbb{I}((d(p_i^c) < \tau_1) \vee (d(p_i^r) < \tau_1)) \cdot d(p_i^r), \quad (6)$$

where $d(p) = \min_{o \in \{o_j \in \mathcal{O} | a_j > \tau_2\}} \|p - o_j\|_2$, and o_j and a_j represent the position and affordance values of object points, and p_i^c and p_i^r represent the hand contact candidates of the initial coarse hand and current refined hand.

- Affordance fingertip loss. The affordance fingertip loss is designed to keep the fingertip positions that are in contact with the affordance area unchanged.

$$\mathcal{L}_{aff-finger} = \sum_i \mathbb{I}(d(q_i^c < \tau_3)) \cdot \|q_i^r - q_i^c\|_2 \quad (7)$$

where q_i^c and q_i^r represent the fingertip position of the initial coarse hand and current refined hand.

- Object Penetration Loss. The object penetration loss \mathcal{L}_{pen} penalizes the depth of hand-object penetration, where

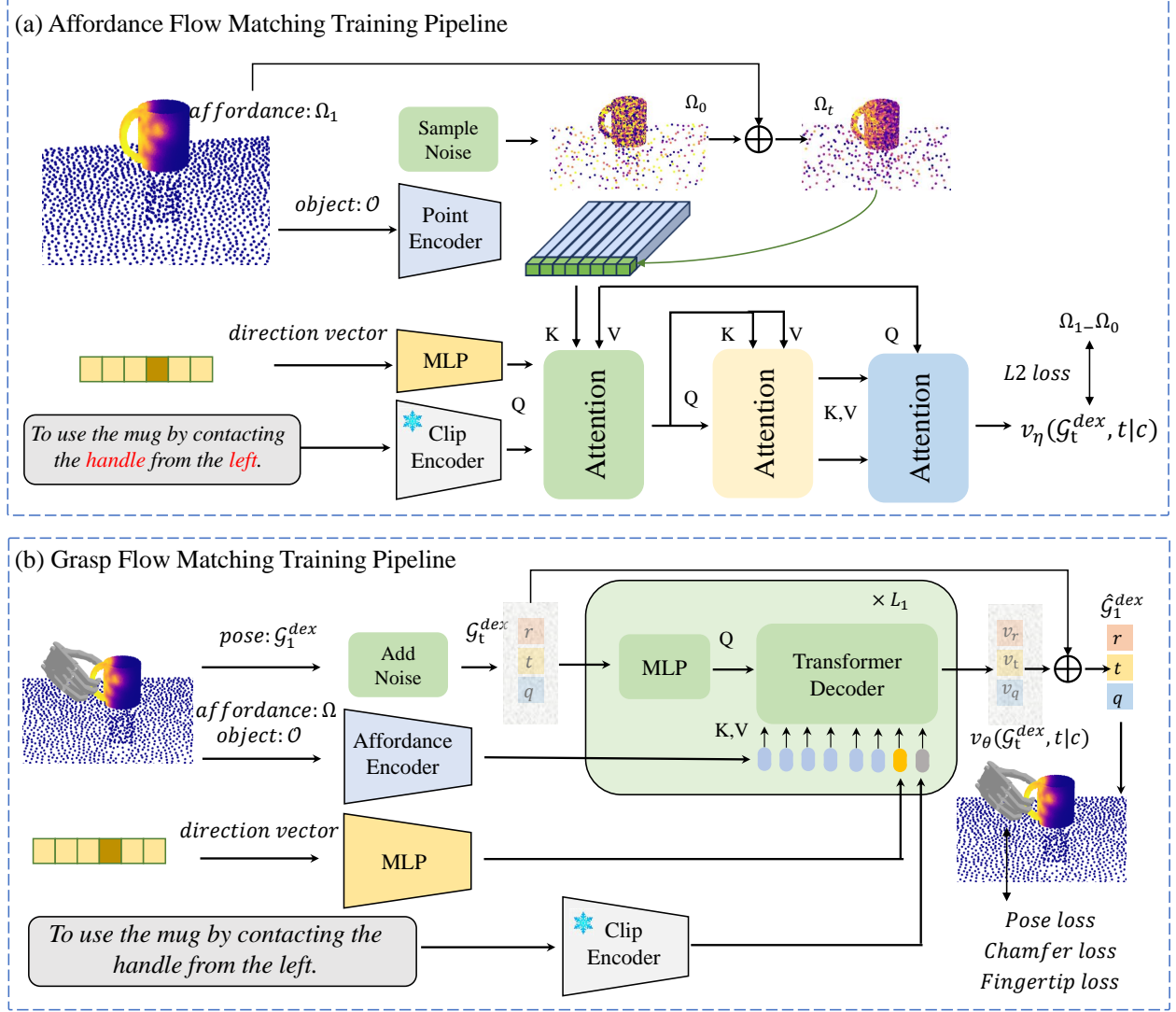


Figure 1. The training pipeline of Affordance Flow Matching and Grasp Flow Matching.

d_i^{sdf} denotes the signed distance from the object point to the hand mesh.

$$\mathcal{L}_{pen} = \sum_i \mathbb{I}(d_i^{sdf} > 0) \cdot d_i^{sdf}. \quad (8)$$

- Self-Penetration Loss. The self-penetration loss \mathcal{L}_{spen} punishes the penetration among the different parts of the hand, where $p^{dex,sp}$ denotes predefined anchor spheres on the hand [19].

$$\mathcal{L}_{spen} = \sum_{i,j} \mathbb{I}(i \neq j) \cdot \max(\delta - d(p_i^{dex,sp}, p_j^{dex,sp})). \quad (9)$$

- Joint Limitation Loss. Given the physical structure limitations of the robotic hand, each joint has designated upper

and lower limits. The joint angle loss penalizes deviations from these limits.

$$\mathcal{L}_{joint} = \sum_i (\max(q_i - q_i^{max}) + \max(q_i^{min} - q_i)). \quad (10)$$

A.4. Pre-understanding by MLLM

We employ Chain-of-Thought (CoT) and visual prompt to enhance the reasoning ability of MLLM as shown in Figure 2. Specifically, the MLLM is asked to reason the following task step by step: recognize the object category, understand the intention, reason the contact part based on intention and category, reason the direction based on the above. Furthermore, we add arrows of different colors in the four direc-

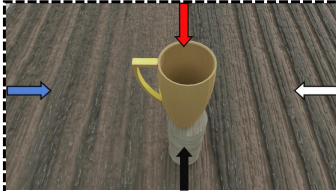

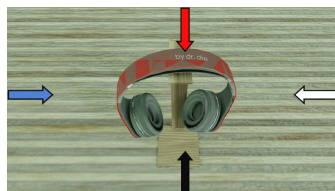




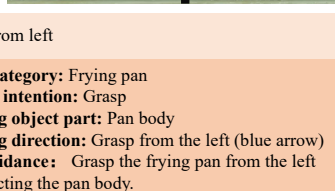


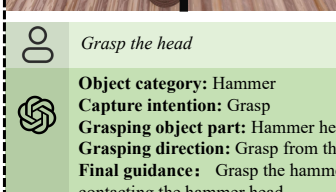


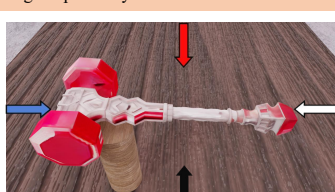

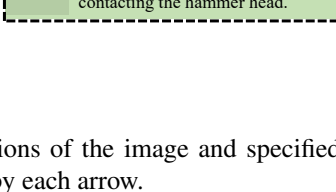


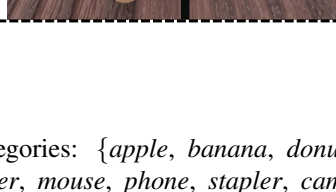

System Prompt : Your task is to guide the robot to execute user commands. Firstly, accept human commands, then analyze the images and output information in a specified format to guide the robot to grasp and manipulate objects.			
Please analyze step by step in the following order: <ol style="list-style-type: none"> 1. What category does the object in the picture belong to 2. What is the intention of grabbing (use/hold/lift/hand over) 3. What part of this object should be grasped (e.g. handle or body) 4. In which direction should this part be grasped (clockwise from red to blue indicates grasping from above, grasping from right, grasping from front, grasping from left). 5. Final guidance: <intention> the <category> from <direction> by contacting <object part>. And output: Object category: {} Capture intention: {} Grasping object part: {} Grasping direction: {} Final guidance: {}			
	 Use it	 Grasp from above	
	 Object category: Mug Capture intention: Use Grasping object part: Handle Grasping direction: Grasp from the left (blue arrow) Final guidance: Use the mug from the left by contacting the handle.	 Object category: Mug Capture intention: Grasp Grasping object part: Mug body Grasping direction: Grasp from above (red arrow) Final guidance: Grasp the mug from above by contacting the body.	
	 Grasp the right ear	 Lift it	
	 Object category: Headphones Capture intention: Grasp Grasping object part: Right ear cup Grasping direction: Grasp from the right (white arrow) Final guidance: Grasp the headphones from the right by contacting the right ear cup.	 Object category: Headphones Capture intention: Lift Grasping object part: Headband Grasping direction: Grasp from above (red arrow) Final guidance: Lift the headphones from above by contacting the headband.	
	 Use it	 Grasp from left	
	 Object category: Frying pan Capture intention: Use Grasping object part: Handle Grasping direction: Grasp from the right (white arrow) Final guidance: Use the frying pan from the right by contacting the handle.	 Object category: Frying pan Capture intention: Grasp Grasping object part: Pan body Grasping direction: Grasp from the left (blue arrow) Final guidance: Grasp the frying pan from the left by contacting the pan body.	
	 Grasp the head	 Use it	
	 Object category: Hammer Capture intention: Grasp Grasping object part: Hammer head Grasping direction: Grasp from the left (blue arrow) Final guidance: Grasp the hammer from the left by contacting the hammer head.	 Object category: Hammer Capture intention: Use Grasping object part: Handle Grasping direction: Grasp from the right (white arrow) Final guidance: Use the hammer from the right by contacting the handle.	

Figure 2. The prompt used in per-understanding stage.

tions of the image and specified the direction represented by each arrow.

B. Zero-shot Datasets Details

B.1. Dataset Splitting

To support our Cross-Category Open-set settings, we divided the dataset into Open Set A and Open Set B by excluding specific samples from the training set. Specifically, the test set of Open Set A consists of 9,688 samples of 808 objects from 11 unseen categories: *{bowl, cylinder, wineglass, headphones, flashlight, pen, wrench, toothbrush, hammer, teapot, scissors}*. The test set of Open Set B consists of 29,744 samples of 568 unseen objects from 10 unseen categories: *{bottle, cup, squeezable, eyeglasses, light-bulb, fryingpan, knife, screwdriver, mug, pincer}*. For one-shot experiments, we add an additional 3,688 samples of

152 objects from 12 categories: *{apple, banana, donut, binoculars, gamecontroller, mouse, phone, stapler, cameras, lotion_pump, power drill, trigger sprayer}*.

B.2. Texture generation

As part of the texture generation, we utilized GPT-4o[7] to generate possible colors and materials for specific object categories. The prompt used for this process was as follows:

Prompt: “Please generate a list of possible colors and materials for the following categories and provide the output in JSON format. The objects are: {binoculars, mug, ...}.”

For objects lacking texture, we employ Paint3D[20] to generate their visual appearance. The positive prompt is constructed using the object’s category name along with a sampled color and material from the predefined possible

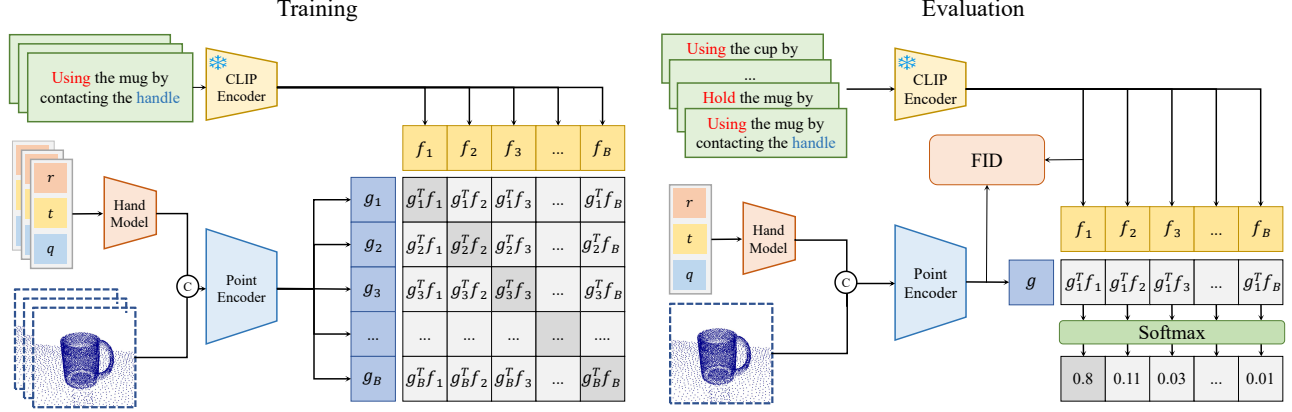


Figure 3. The training pipeline of evaluation encoder (left). The evaluation of FID and R-Precision.

lists, while the default negative prompt is utilized to guide the generation process.

Prompt: “<category>, <sampld color>, <sampld material>, high quality, clear”

Negative prompt: “strong light, Bright light, intense light, dazzling light, brilliant light, radiant light, Shade, darkness, silhouette, dimness, obscurity, shadow”

B.3. Scene Reconstruction

The details of the construction of our scene dataset are illustrated in Figure 4. We utilize BlenderProc [1] to create a tabletop scene and project the grasping poses from the full-model dataset into the scene. After performing a gravity check on the objects and a collision test between the objects and the robotic hand, we record the grasp parameters in the scene coordinate system. At the same time, we capture RGB and depth images from the scene cameras. Finally, we construct the scene point cloud by concatenating and downsampling the partial point clouds obtained from the depth images.

In our tabletop scene, objects are lifted using three types of shelves. The first is a cylindrical base, which is used for the majority of categories. The second is a smaller shelf, consisting of two circular planes connected by a rod, designed for smaller categories such as pens and toothbrushes. The third is a headphone-specific shelf, used exclusively for headphones.

The poses of five RGB-D cameras are determined based on the center of the object’s bounding box. One camera is positioned directly above the center of the bounding box. The other four cameras are placed at the four sides of the bounding box, with a side offset equal to half the length of the bounding box’s longest edge and a height offset equal to one-fourth of the bounding box’s longest edge.

C. Experiments Details

C.1. Evaluation Details

- Training pipeline of evaluation encoder. The training pipeline of our Intention Evaluation Model is illustrated in Figure. 3. The model takes as input the hand parameters \mathcal{G}^{dex} , the object point cloud $\{o_i\}_{i=1}^M$, and the language guidance \mathcal{L} . Initially, the hand model generates the hand point cloud $\{h_i\}_{i=1}^N$, which is then concatenated with the object point cloud $\{o_i\}_{i=1}^M$ to form the combined hand-object point cloud $\{p_i\}_{i=1}^P$.

A PointNet++ encoder and a CLIP encoder are subsequently employed to map $\{p_i\}_{i=1}^P$ and \mathcal{L} into the feature space, producing the grasping feature \mathbf{g} and the guidance feature \mathbf{f} , respectively. The Information Noise-Contrastive Estimation (InfoNCE) Loss[12] is then applied to maximize the cosine similarity between paired features while minimizing it between unpaired features. The InfoNCE Loss is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\sum_{j=1}^B \mathbb{I}[\text{match}(i, j)] \exp(\text{sim}(\mathbf{g}_i, \mathbf{f}_j)/\tau)}{\sum_{j=1}^B \exp(\text{sim}(\mathbf{g}_i, \mathbf{f}_j)/\tau)}, \quad (11)$$

where $\text{match}(i, j)$ indicates that the object category and action of the i -th grasp match those of the j -th grasp (including the case where $i = j$). Here, $\text{sim}(\mathbf{g}, \mathbf{f})$ denotes the cosine similarity between features \mathbf{g} and \mathbf{f} , τ is a temperature parameter, and B is the number of samples in the batch. The loss encourages high similarity for matched pairs $(\mathbf{g}_i, \mathbf{f}_j)$ and low similarity for unmatched pairs $(\mathbf{g}_i, \mathbf{f}_j)$ where $i \neq j$ and $\text{match}(i, j)$ does not hold.

To ensure the model’s ability to distinguish between different actions within the same category, we design a custom sampler for the training data loader. The sampler randomly selects 8 different categories and randomly chooses

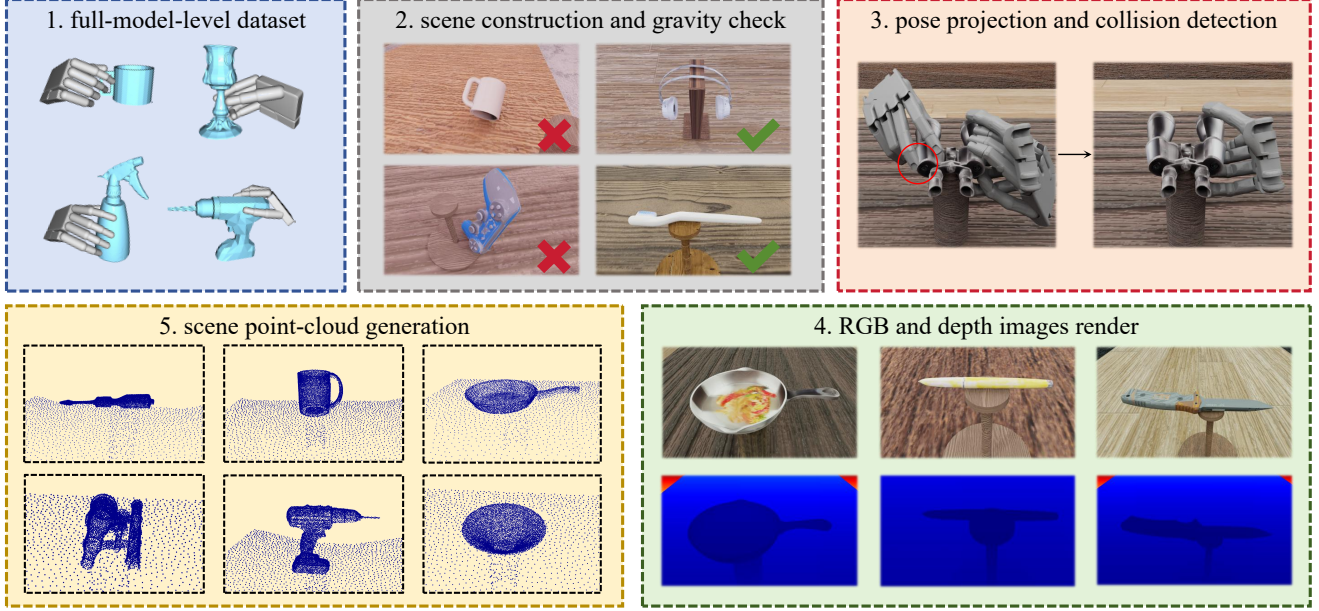


Figure 4. The construction of our scene dataset.

$\frac{B}{8}$ grasps for each category. This process is repeated until the number of remaining categories in the dataset is fewer than 8.

- **R-precision** When computing R-Precision during evaluation, we first randomly select 3 grasp guidance samples from the dataset that belong to the same category but involve different actions, along with 28 guidance samples from other categories. Next, we generate grasp features and a batch of guidance features using the same method as in the training stage. We then compute the cosine similarity between the grasp feature and each guidance feature. Finally, we determine the rank of the matched guidance sample based on these similarity scores.

- **Fréchet Inception Distance (FID)**. This metric is widely used in generative tasks [2, 3] to assess the similarity between the generated and ground truth distributions. For our evaluation, we compute the FID using features extracted from the object and hand point clouds.

- **Chamfer Distance (CD)**. Chamfer distance, denoted as CD , measures the discrepancy between the predicted hand point clouds and the nearest ground truth targets, providing a consistency measure from the perspective of hand positioning. The closest targets are the ground truth grasps that share the same intended grasp and are most similar to the generated grasp. This methodology follows the approach in [18].

- **Grasp Success Rate**. The success rate of grasps is evaluated within the Isaac Gym simulation environment. To mimic the forces exerted by a dexterous hand in real-world scenarios, each finger is contracted towards the object. A

grasp is considered successful if it can resist at least one of the six gravitational directions, indicating it can sustain a stable hold on the object.

- **Mean Q_1 Metric**. The Q_1 metric quantifies the smallest wrench capable of destabilizing a grasp. We follow the guidelines in [17], where the contact threshold is set at 1 cm and the penetration threshold is set at 5 mm. Any grasp with a maximal penetration depth exceeding 5 mm is considered invalid, and its Q_1 value is set to 0 before averaging the results.

- **Maximal Penetration Depth**. This metric measures the greatest penetration from the object’s point cloud to the hand mesh during a grasp.

- **Diversity**. To assess the diversity of generated grasps, we calculate the standard deviations of translation (δ_t), rotation (δ_r), and joint angles (δ_q). We generate eight samples for each intended grasp under identical input conditions, and each sample is then sent for refinement through the quality component. For calculation purposes, δ_r and δ_q are converted into Euler angles (in degrees), and δ_t is measured in centimeters.

C.2. Reproduction of SOTA method

We reproduce SOTA methods on our Open Set dataset using the same encoder structure to ensure fair comparison. Specifically, we reimplement ContactGen [11], Contact2Grasp [10], GraspCAVE based on [15], SceneDiffuser [5], and DexGYS [18]. To introduce language information, we use an identical CLIP language encoder. For GraspCAVE, ContactGen and Contact2Grasp, we concatenate the

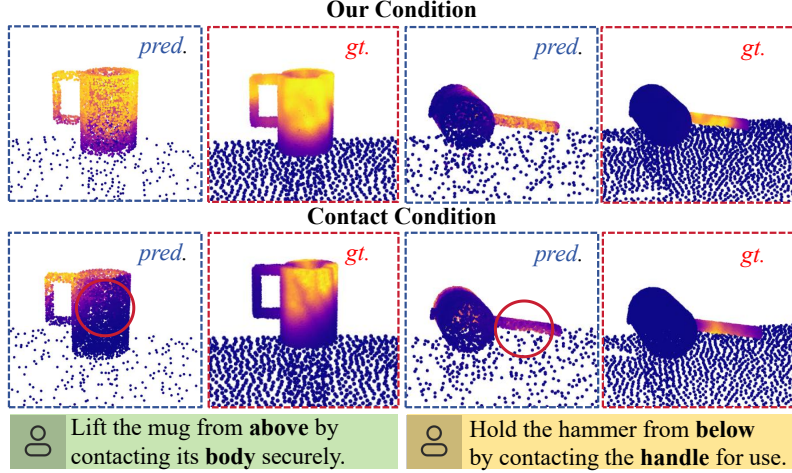


Figure 5. The visualization of our affordance and contact map for prediction and the ground truth.

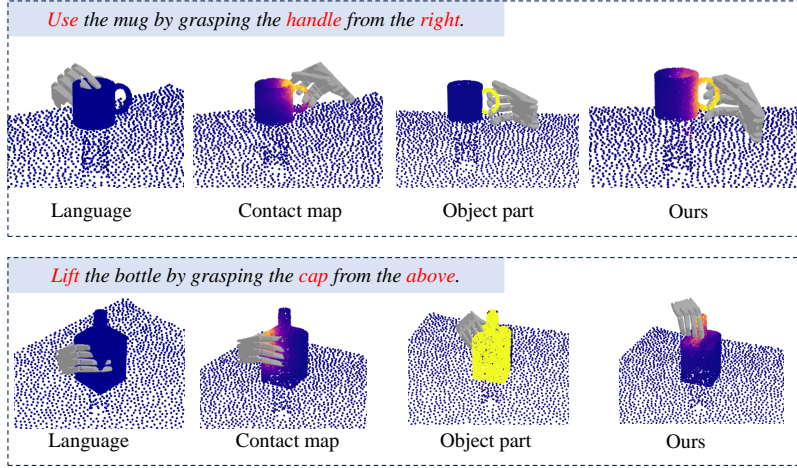


Figure 6. The visualization of generated grasps under different condition with *only language*, *contact map*, *object part* and *ours*

language feature, object feature, and latent feature to send to the decoder. For SceneDiffuser, we concatenate the language and object features as the model condition. The input language guidance of all methods are identical as the MLLM’s output for fair comparison.

D. More Experiments

D.1. Effectiveness of Flow matching model

The results in Table 1 and 2 shows the performance of Our flow matching based model with the diffusion based model. It is noticed that our model surpass the diffusion based model with higher performance and faster inference speed. In addition, the introduced of the affordance flow matching only increased the time by 0.075 second compared to the baseline.

D.2. Experiments of Affordance

Qualitative Experiments. As shown in Figure 5 and 6, our generalizable-instructive affordance is more generalizable and instructive than other representations on open set. The language condition, without low level affordance condition struggles to generalize to unseen categories. The contact map is too fine-grained to generalize and the object part is too coarse to provide effective guidance.

Quantitative Experiments. We conduct an experiment to prove the proposed affordance are better than contact map in generalization, as shown in Tab. 3. The Point IOU metric is used for measure the consistency of prediction and the ground truth. The performance of contact maps drops markedly in testing, compared with affordance.

	<i>Open Set A</i>					<i>Open Set B</i>				
	<i>FID</i> ↓	<i>CD</i> ↓	<i>Top1</i> ↑	<i>Q1</i> ↑	<i>Pen.</i> ↓	<i>FID</i> ↓	<i>CD</i> ↓	<i>Top1</i> ↑	<i>Q1</i> ↑	<i>Pen.</i> ↓
DDIM [16]	0.247	4.13	0.455	0.0167	0.541	0.211	3.66	0.520	0.0167	0.542
DDPM [4]	0.254	3.97	0.461	0.0283	0.463	0.204	3.50	0.516	0.0128	0.602
Flow Matching	0.242	3.79	0.480	0.0240	0.501	0.176	2.76	0.538	0.0150	0.612

Table 1. The comparison of Flow Matching with Diffusion models with DDPM and DDIM.

	AFM	GFM	DDIM	DDPM
Time	0.075 ± 0.006	0.233 ± 0.022	0.235 ± 0.013	1.14 ± 0.099

Table 2. The inference time of Affordance Flow Matching (AFM) (first column); Grasp Flow Matching (second column); Grasp generation with DDIM and DDPM (The third and fourth columns).

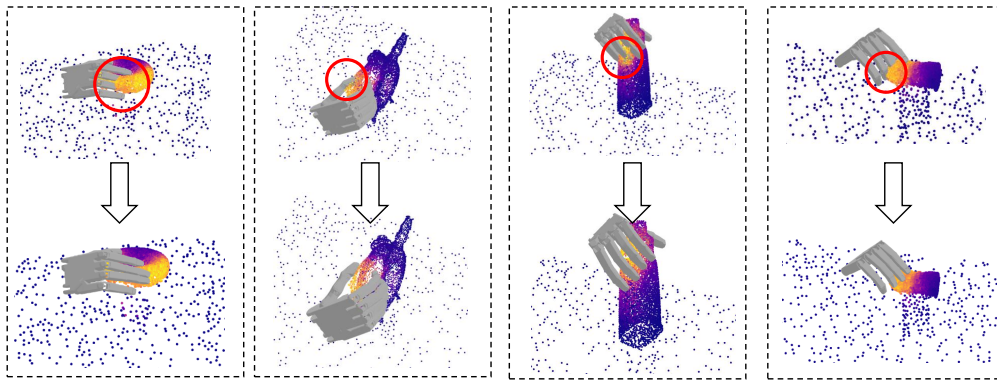


Figure 7. Visualization of grasps before and after affordance-guided grasp optimization.

D.3. Qualitative Experiments of Affordance-guided Optimazation

To verify the effectiveness of the affordance-guided optimization, we offer more qualitative results. Figure 7 shows the grasping situations before and after optimization. It clearly shows that the QGC can prevent the grasp from penetrating the object and keep in line with the original intention.

D.4. Extension to Dexterous Manipulation

We extend our methods to the manipulation task by cooperating the grasp ability of our framework with task planning ability of MLLM and the perception ability of computer vision foundation model to achieve dexterous manipulation task, like pouring water, following ReKep [6]. Specially, the MLLM first generates the relational keypoint constraints, then obtain an action solution by optimization solver. Then we employ our framework to generate a stable dexterous grasping and execute manipulation action according to the action solution. The visualization results can be found in Figure 10.

E. Real World Experiments Details

Experimental Environment Figure 8 shows the settings of our real world experiments. The experiments are conducted on Leap hand, a Kinnova Gen3 6DOF arm and a Kinova original wrist RGB-D camera. We collect several daily life objects to evaluate the open-set generalization in real world.

Experiment Pipeline To obtain the scene point clouds, we set the robotic arm to record partial point clouds along a specified trajectory, then fuse the local point clouds into a global point cloud and down-sample them to obtain the sampled point clouds, as shown in Figure 9. Next, the RGB image and user commands are fed into MLLM to obtain the guidance. The guidance and scene point clouds are then fed into our framework to sequentially generate affordance and grasp poses. During the grasp execution phase, we first move the robotic arm to the vicinity of the target, and then simultaneously control both the robotic arm and the dexterous hand to perform the grasp. The visualization can be found in Figure 11.

Point IOU \uparrow	Proposed Affordance		Contact Map	
	Train	Test	Train	Test
Set A	0.597	0.508	0.576	0.359
Set B	0.602	0.502	0.624	0.315

Table 3. Generation performance of the proposed affordance and contact map.

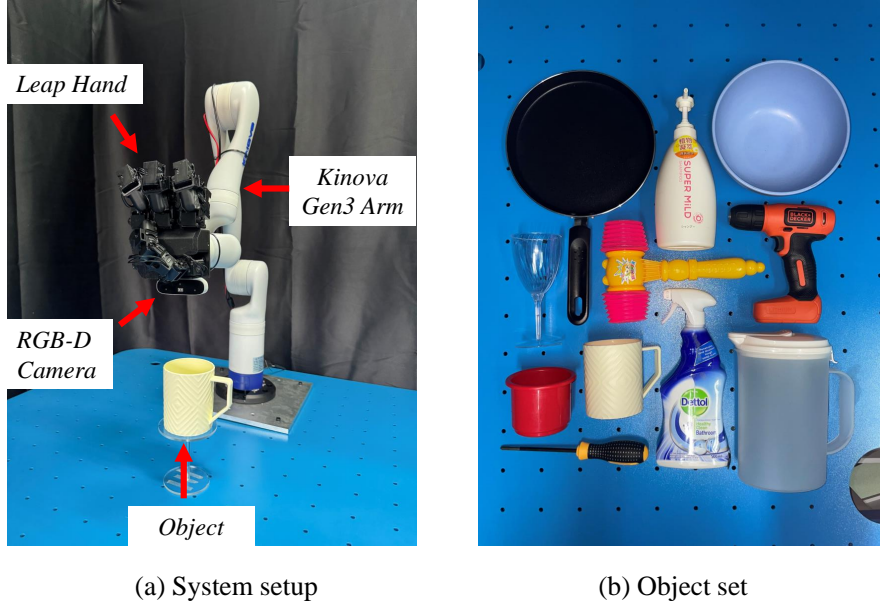


Figure 8. The illustration of our real world experiment settings.

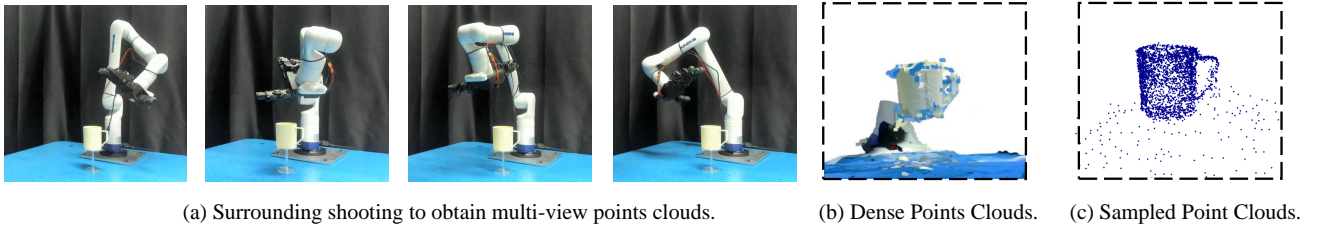


Figure 9. The visualization of the pipeline to obtain scene point clouds.

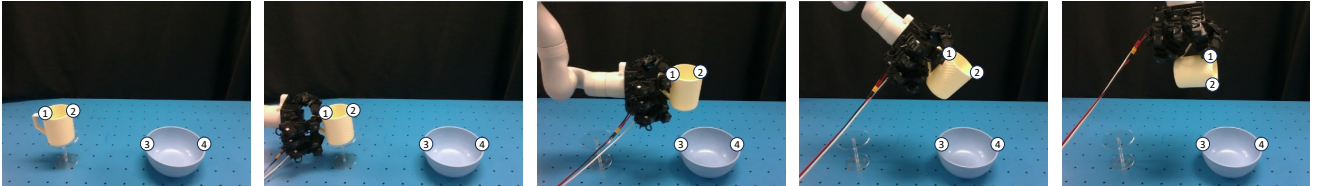


Figure 10. Visualization of the application of our method to the manipulation task.

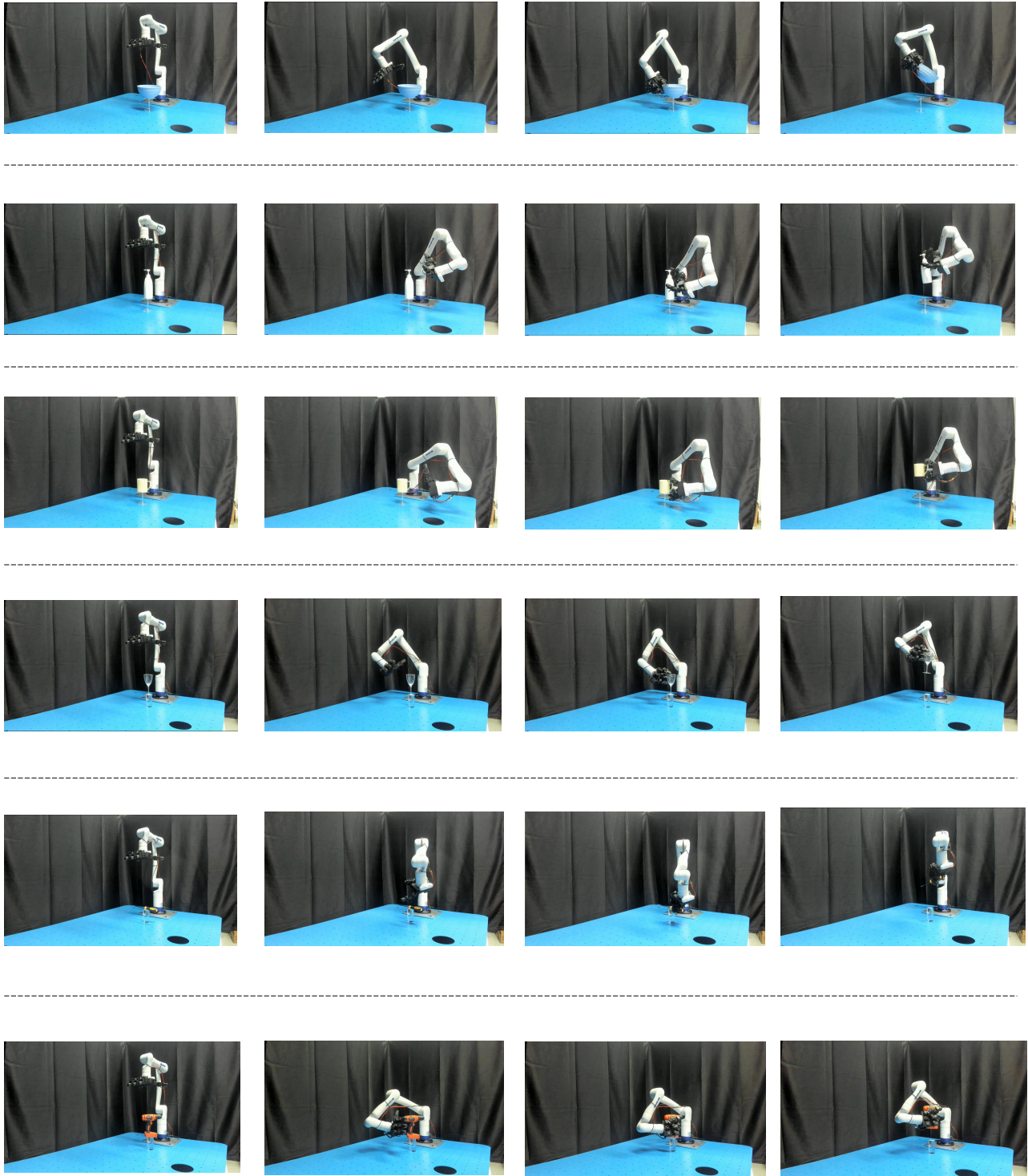


Figure 11. The visualization of motion in real world experiments.

References

- [1] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Wendelin Knauer, Klaus H Strobl, Matthias Humt, and Rudolph Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023. [4](#)
- [2] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5152–5161, 2022. [5](#)
- [3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. [5](#)
- [4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. [7](#)
- [5] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023. [5](#)
- [6] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. [7](#)
- [7] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. [3](#)
- [8] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. [1](#)
- [9] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021. [1](#)
- [10] Haoming Li, Xinzhuo Lin, Yang Zhou, Xiang Li, Yuchi Huo, Jiming Chen, and Qi Ye. Contact2grasp: 3d grasp synthesis via hand-object contact constraint. *arXiv preprint arXiv:2210.09245*, 2022. [5](#)
- [11] Shaowei Liu, Yang Zhou, Jimei Yang, Saurabh Gupta, and Shenlong Wang. Contactgen: Generative contact modeling for grasp generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20609–20620, 2023. [5](#)
- [12] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. [4](#)
- [13] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. [1](#)
- [14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. [1](#)
- [15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. [5](#)
- [16] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. [7](#)
- [17] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11359–11366. IEEE, 2023. [5](#)
- [18] Yi-Lin Wei, Jian-Jian Jiang, Chengyi Xing, Xian-Tuo Tan, Xiao-Ming Wu, Hao Li, Mark Cutkosky, and Wei-Shi Zheng. Grasp as you say: Language-guided dexterous grasp generation. *Advances in Neural Information Processing Systems*, 37:46881–46907, 2025. [5](#)
- [19] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023. [2](#)
- [20] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4252–4262, 2024. [3](#)