

A. Implementation Details of Energy-based Optimization

A.1. Preliminary: the Force-Closure property of dexterous grasp poses

The concept of **force closure** is first proposed in [6] as a physics-based property desirable for multi-contact grasp poses on objects. Under the assumption of the Coulomb friction model [6], force closure is a binary metric indicating whether a grasp pose can resist external wrenches applied to the object and hold the object in a static status:

Definition of Force Closure

If a set of contact points $\{C_m, n_m\}$ in a Coulomb friction model is force closure, where C_m are the positions of contact and n_m are the contact normals pointing to the inside of object, then for any external force F applied on the object, there exists contact forces $\{f_m\}$ that lies within the friction cones correspond to $\{C_m, n_m\}$ and cancel out the external force.

We refer the readers to the paper [6] for a rigorous formulation and more technical details. The concept of force closure offers a handy metric to check whether a dexterous grasp pose is stable. However, it is not clear from the definition **how we can optimize a dexterous grasp pose towards forming a force closure grasp**. Many existing works explore approximation methods and formulate force closure into objective functions of optimization problems [2, 15, 19]. In this paper, we follow [14] and base our method on the differentiable force closure (DFC).

A.2. Preliminary: Differentiable Force Closure (DFC)

[14] first proposes a differentiable force closure estimator (DFC) as an energy term applicable to gradient-based optimization for dexterous grasp synthesis. Given a collection of n contact force vectors, which is approximated by n contact normal vectors $c \in \mathbb{R}^{3n}$ at n positions $x \in \mathbb{R}^{3n}$, DFC encourages them to form a force closure over the grasped shape. Formally, DFC is expressed as

$$E_{fc} = \|Gc\|_2 \quad (1)$$

where

$$c = [c_1, \dots, c_n] \in \mathbb{R}^{3n} \quad (2)$$

$$G = \begin{bmatrix} I_{3 \times 3} & \dots & I_{3 \times 3} \\ [x_1]_{\times} & \dots & [x_n]_{\times} \end{bmatrix} \in \mathbb{R}^{6 \times 3n} \quad (3)$$

$$[x_i]_{\times} = \begin{bmatrix} 0 & -x_i^{(z)} & x_i^{(y)} \\ x_i^{(z)} & 0 & -x_i^{(x)} \\ -x_i^{(y)} & x_i^{(x)} & 0 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (4)$$

$$x_i = (x_i^{(x)}, x_i^{(y)}, x_i^{(z)}) \in \mathbb{R}^3$$

Intuitively, the matrix G , named the *grasp matrix*, transforms a set of contact forces in unit magnitude c into net external force and torque applied to the object. As analyzed in [2, 14], a small value of $\|Gc\|_2$ indicates the set of contact forces effectively cancel out each other without considering the effect of friction, which suggests the contact force vectors span the entire wrench space and therefore can potentially counteract any external wrench applied to the object. As a result, optimizing with E_{fc} encourages synthesizing force-closure grasp poses.

By using surface normal vectors to approximate contact forces, DFC assumes equal magnitude contact forces at all contact points. For more details, please refer to [14].

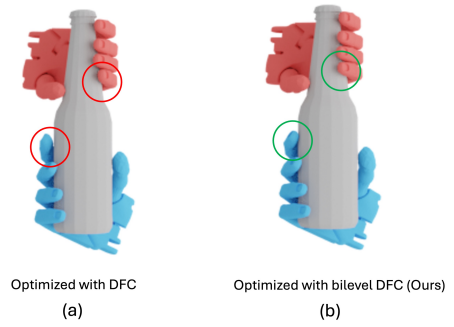


Figure 1. **Visualization of LP-based DFC vs DFC.** **Left:** DFC generates artifacts such as tilted finger and drifted contacts, **high-lighted in red**. **Right:** LP-based DFC generates more natural poses that are compliant with the geometry of the object, **highlighted in green**. The grasp poses are optimized from identical initializations and only differ by using DFC or LP-based DFC for optimization.

A.3. LP-based DFC

As analyzed in DexGraspNet 2.0 [24], the assumption of equal-magnitude contact force in DFC limits the quality of synthesized grasp poses. To this end, we use linear programming that takes contact force magnitude into account. At each timestep, we first fix the hand pose and solve the optimal contact forces applied to each contact position, such that the net wrench applied to the object is minimized:

$$\begin{aligned} P &= \min_{\mathbf{f}} \|G(\mathbf{f} \odot c)\|_2 \\ s.t. \max_i (\mathbf{f})_i &= 1 \\ (\mathbf{f})_i &\geq 0, i = 1, 2, \dots, n \end{aligned} \quad (5)$$

Then we use the net wrench P and corresponding contact forces \mathbf{f} as hyperparameters to rescale the DFC energy

when the current pose is already stable. Specifically, we check whether the total wrench is small enough and each contact point applies non-trivial force. If the current pose is not stable, we optimize with the original DFC to search for stable poses efficiently. We empirically find that grasps synthesized with LP-based DFC are more natural than those using DFC.

$$E_{FC} = \begin{cases} \|G(\mathbf{f} \odot \mathbf{c})\|_2, & \text{if } P < \tau_{FC} \text{ and } \min_i(\mathbf{f})_i \geq \tau_f \\ \|G\mathbf{c}\|_2, & \text{otherwise} \end{cases} \quad (6)$$

A qualitative comparison between LP-based DFC and the original DFC is visualized in Fig. 1. Starting from the same initialization pose, we optimize the grasp poses with DFC and LP-based DFC, respectively, with identical hyperparameters and all other energy functions for optimization unchanged. Grasp poses synthesized with DFC suffer from artifacts of tilted fingertips and broken contacts. Under the assumption that each contact point applies equal contact pose, the DFC energy is not minimized at a natural grasp pose of the thumb opposing the rest 4 fingers. Therefore, minimizing with respect to DFC encourages the hand to drift away from the natural pose in this scenario. On the other hand, the LP-based DFC takes contact magnitude into account and produces grasp poses more natural and compliant to the geometry of the object.

A.4. Regularization Energy

We implement several regularization energies to counteract undesirable behaviors in the process of gradient-based optimization.

First, we would like to avoid both hand-object penetration and hand self-penetration in the optimization process. To this end, we incorporate the built-in implementation of penetration energy (denoted E_{pen}) and self-penetration energy (denoted E_{spen}) in the CuRobo [17] robotics library.

Second, we would like to prevent hand joint angles from going outside the limits. We incorporate the built-in implementation of joint limit energy in CuRobo [17] as well.

Third, in order to avoid synthesizing dexterous grasps in twisted poses, we encourage the hand-object contact points to align with the front side of the hand meshes. We use a cosine-similarity energy function

$$E_{dir} = \sum_{i=0}^n (1 - c_i \cdot N_i) \quad (7)$$

where n is the total number of contact points on the hand, c_i is the i -th contact normal vector on the object surface pointing inwards into the object, and N_i is the normal vector of the front surface of the i -th contact link, pointing outwards from the hand link mesh.

In summary, the complete regularization energy is formulated as

$$E_{reg} = \omega_{limit} E_{limit} + \omega_{pen} E_{pen} + \omega_{spen} E_{spen} + \omega_{dir} E_{dir} \quad (8)$$

B. Implementation Details of Object Preprocessing

The object assets in our DexGraspNet 3.0 dataset are pre-processed and filtered from the entire Objaverse [4] dataset, which encompasses over 800k 3D assets covering diverse object categories.

We first filter the assets by querying GPT-4o [1]. Following SoFar [16], we concatenate images rendered from six orthogonal views of each object, and prompt GPT-4o with true-or-false queries of the following five requirements:

- Whether the object is a scene or a collection of disconnected objects.
- Whether the object has single white or grey colors.
- Whether the object has a ground/tabletop plane.
- Whether the 3D model is a refined, well-constructed mesh without defects.
- Whether the object represents a recognizable and meaningful 3D object.

The detailed prompt used is identical to Fig.15 in [16]. After filtering out undesired objects, we process the rest objects by extracting .obj meshes using Trimesh [3], and then obtaining a convex-decomposed watertight mesh with ManifoldPlus [9] and CoACD [21](with threshold=0.4). Low-quality objects that fail watertight remeshing, as well as too complex objects that generate more than 200 convex pieces are discarded.

We use SAMesh [18] to segment the remaining valid meshes into semantically meaningful parts. The segmented meshes are colored with segmentation masks and rendered from 6 orthogonal views. The multi-view images are then concatenated and fed into GPT-4o with Set-of-Marks prompting to obtain the semantic captions of each part. Visualization of sample segmented meshes together with the semantic part label is provided in Fig. 2. The prompt used for part name labeling is given in Fig. 10.

Finally, we ask GPT-4o to output a reasonable size of each object and rescale the objects accordingly. In order to make sure the semantically meaningful parts of each object are in a graspable size, we clip the diagonal length to the range [20, 50]cm. The prompt used to query object size is given in Fig. 11.

C. Implementation Details of Hand Pose Initialization

This section provides implementation details of how we align the initial hand poses with the object part geometry.



Figure 2. Visualization of part segmentation and GPT generated part annotations.

Given an object segmented into semantic parts, we sample a batch of 5000 initial hand poses around each part and perform gradient-based optimization to synthesize grasp poses aligned with the specified part. We first extract oriented bounding boxes of each part (Sec. C.1). The oriented bounding boxes give geometric priors about the size, shape and orientation of object parts. The spatial relation among bounding boxes of different parts gives hints about the entire object’s structure [10, 16]. Leveraging these geometric cues, we align the initial position and orientation of the palm of the hands with object parts. Lastly, to synthesize diverse grasp poses, we jitter the initial palm poses and handcraft two sets of initial hand joint poses together with candidate contact points on the hand in Sec. C.3, resulting in the **Ours-Wrap** and **Ours-Pinch** splits of the DexGraspNet 3.0 dataset.

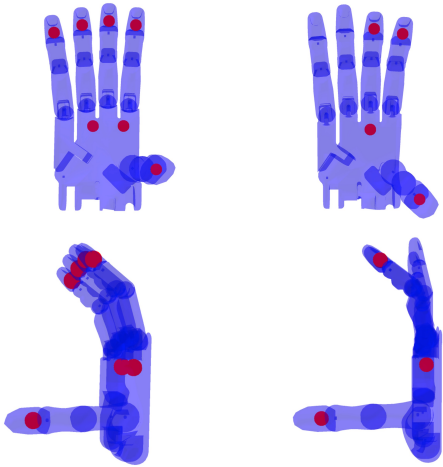


Figure 3. Visualization of initial hand joint positions and contact candidates.

Split Name	Object	Grasps	Captions	Contact Candidates
Ours-Wrap	169k	103M	103M	7
Ours-Pinch	139k	67M	67M	4
Total	174k	170M	170M	-

Table 1. Statistics of DexGraspNet 3.0 dataset in splits.

C.1. Oriented Part Bounding Box Generation

We generate the bounding box of each object part in order to quantify the geometry and orientation of each part. In specific, we sample 256 directions using Fibonacci lattices [5, 7] as the x-axis, and for each direction, compute the in-plane rotation for generating the tightest part bounding box following [8]. We select the minimal volume box among these 256 candidates as the oriented bounding box of the object part.

We refer to the axis directions of the oriented part bound-

ing box as **part principal directions**. The part principal directions build a local frame on the object part that indicates its orientation.

C.2. Aligning Initial Palm Poses With Part Geometry

Depending on the direction, length and inter-part relation of part principal directions, we classify all object parts in the DexGraspNet 3.0 dataset into 4 categories, which are detailed as follows. For each category, we use specific strategies to sample grasp points on the surface of the part and align initial palm poses with the part’s principal directions. Although our categorization is not rigorous nor exhaustive, we observe that most functional parts of objects are similar in shape and are designed to be grasped in a few specific styles. Moreover, as pointed out in [20], the quality of synthesized grasp poses in gradient-based optimization is very sensitive with respect to initialization poses. By aligning hand initial poses with human-inspired heuristic rules, we inject a strong human-alignment prior into the optimization process, which greatly helps us to synthesize more natural and semantically distinguishable grasp poses. As visualized in Fig. 9, grasp poses synthesized with our meticulously designed initialization are more natural than those initialized in random poses.

We categorize the object parts into 4 categories: **lid-like**, **disk-like**, **L-shaped** and **shaft-like**;

Lid-like part is typical for knobs, lids of containers and some types of handles. We categorize a part as a lid-like part if the following conditions are met:

- At least 4 corners of the part bounding box are inside the bounding box of another part, which indicates the current part is “embedded inside” this other part.
- Any 4 corners at the same end of the longest axis of the part bounding box should not be inside a single part bounding box. This excludes the case that the part is a shaft.



Figure 4. Lid-like part

For lid-like parts, we identify the part’s principal direc-

tion as the bounding box axis that points outside of the other part it is embedded in. We sample grasp points on the surface area where the surface normal aligns with the principal part direction. We align the palm to retreat from each grasp point and be perpendicular to the part principal direction and jitter by rotating the hand along the part principal direction for 24 uniform angles. One example of initialization with lid-like parts is visualized in Fig. 4.

Disk-like part is typical for the supporting surfaces of objects, such as base of a lamp, the seat of a chair, and the tabletop surfaces. We categorize a part as a disk-like part if the following conditions are met:

- The second-shortest axis is at least 3 times longer than the shortest axis, which indicates the part is flat.
- None of the corners of the part’s bounding box is inside another part, which indicates the current part is protruding.

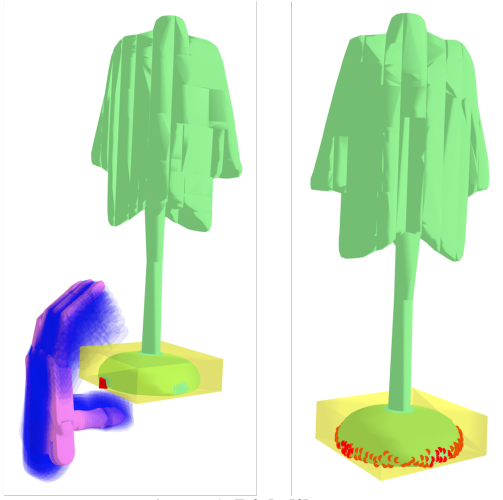


Figure 5. **Disk-like part**

For disk-like parts, we identify the shortest axis of its bounding box that points into an adjacent part as the part principal direction. We sample grasp points on the surface area where the surface normal is perpendicular to the part principal direction. We align the palm to retreat from each grasp point with the front face of palm perpendicular to the surface normal of the grasp point and the up direction of palm align with the principal direction. We jitter the palm pose by slightly rotating it sideways, as shown in Fig. 5.

L-shaped part is typical for thin object parts designed for human-object interaction, such as the headband of headphones, the seats of vehicles and the handle of kettles. We categorize a part as an L-shaped part if any corner section of the part’s bounding box does not intersect with the mesh of the part.

For the L-shaped part, we identify the part principal direction as the direction pointing from the center of part bounding box to the middle of the bounding box edge cor-

responding to the missing corner section. We sample grasp points on the surface area where the surface normal is perpendicular to the principal part direction. We align the palm to retreat from each grasp point with the front face of the palm perpendicular to the surface normal of the grasp point and the up direction of the palm aligns with the direction pointing from the grasp point to the center of the part’s bounding box. We jitter the palm pose by slightly rotating it sideways, as shown in Fig. 6.

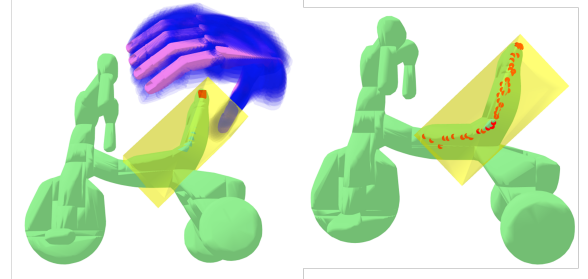


Figure 6. **L-shaped part**

Shaft-like part is typical for handles of tools and connecting bars of long objects such as lamps and barbells. Object parts that are not categorized into any of other types are categorized as shaft-like parts by default, because we empirically find the initialization strategy for shaft-like part is relatively more robust to uncommon geometries.

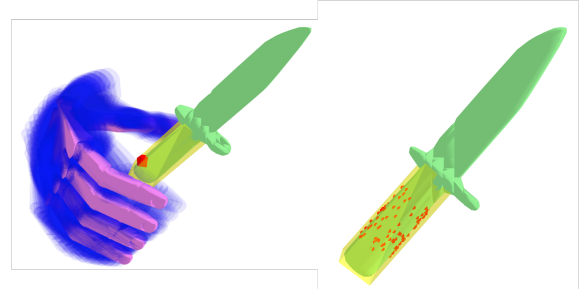


Figure 7. **Shaft-like part**

For shaft-like part, we identify the part principal direction as the longest axis of part bounding box aligned with the direction from part bounding box center to object bounding box center. We sample grasp points on the surface area where the surface normal is perpendicular to the part principal direction. We align the palm to retreat from each grasp point with the front face of palm perpendicular to the surface normal of the grasp point and the principal direction of palm aligned with the part principal direction. We jitter the palm pose by slightly rotating it sideways, as shown in Fig. 7.

C.3. Hand Joint Pose and Collection of Contact Points

Following [24], we handcraft initial joint angles and contact candidates of the hand, as visualized in Fig.3. Contact candidates refer to the labeled points on the hand mesh, with which we calculate the distance energy E_{dis} and encourage these points to make contact with the object. In **Ours-Wrap** split, we set contact candidates on all 5 fingertips as well as on the palm. Grasp poses synthesized in In **Ours-Wrap** split usually wrap the object part in the hand. In **Ours-Pinch** split we set contact candidates on the thumb, forefinger, middlefinger and palm. Grasp poses synthesized in **Ours-Pinch** split are dominantly pinch grasps, which are more suitable for grasping tiny object parts. For each part on each object, we generate a batch of 5000 initialization poses for each of **Ours-Wrap** and **Ours-Pinch** split, respectively for the optimization process.

D. Camera Views Used to Render the Tabletop Dataset

We visualize the camera poses used to render the DexGraspNet 3.0 tabletop dataset in Fig. 8. We place the object to grasp onto the table with stable poses collected from simulation, and manually set the translation in the x and y direction to zero. We evenly sample camera poses in the circle 80cm away from the table frame center and look down at 45 degrees in the yaw, looking at the origin.

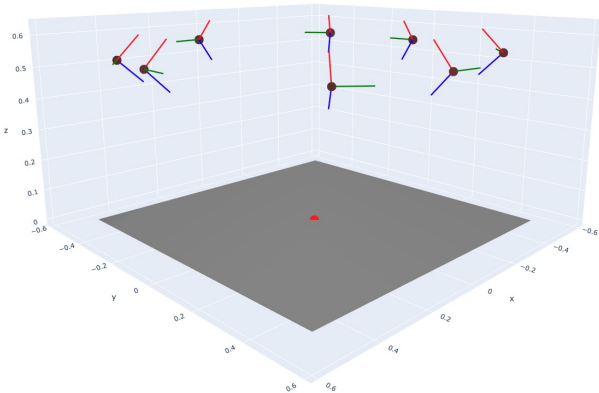


Figure 8. **Visualization of camera poses used to render the tabletop dataset.** We place the object on the table with translation in the x and y direction zeroed. The red dot represents the place where the object is placed. We sample camera poses evenly in the circle 80cm away from the table frame center and look down with 45 degrees in the yaw, looking at the origin. The camera frame (+x,+y,+z) of each camera pose is visualized.

E. More Experiments

E.1. Grasp Quality of DexGraspNet 3.0 Dataset

Grasp poses in the DexGraspNet 3.0 dataset have less severe penetration and self-penetration, because we performed simulation-based penetration checking and filtered the undesirable poses. The poses in DexGraspNet 3.0 also demonstrate comparable stability quality compared against existing dexterous grasp datasets, with comparable Q1 statistics. Different from existing datasets, DexGraspNet 3.0 addresses the task of part-aligned grasp synthesis that emphasizes semantic alignment. The Q1 stability metric of DexGraspNet 3.0 is not expected to be higher than that in power grasp datasets, in which stability is the only goal of data synthesis.

Method	Scale↑	Pen↓	SPen↓	Q1↑
DexGraspNet [20]	1.32M	13.5	0.93	0.114
DexGYS [22]	50k	3.32	-	0.074
SemGrasp [12]	50k	1.1	-	-
Multi-GraspLLM[11]	120k	7.1	-	0.091
Ours-Wrap	103M	1.75	0.19	0.085
Ours-Pinch	67M	1.42	0.22	0.067

Table 2. **Evaluation of the dataset.** Pen-d and SPen-d are in mm.

E.2. More Ablation Experiments

As shown in Table 3, we also conduct more ablation experiments on grasp style. We find that the wrap grasp consistently outperforms the pinch grasp in both success rate and part graspability across all datasets, indicating greater stability and effectiveness in interacting with object parts. While the pinch grasp offers more flexibility and maneuverability, it struggles with stability, leading to lower performance. Both grasp styles see a drop in performance on unseen objects, highlighting challenges in generalization. This suggests a trade-off between stability and precision, where the wrap grasp is better suited for secure, robust grasps, while the pinch grasp may be preferable for delicate or precision-based tasks.

As shown in Table 4, the flow-matching paradigm is much better than traditional DDPM and DDIM diffusion-based action denoise modules. The simple flow-matching denoise module facilitates easier learning of pose generation.

Data	LVIS-Seen		Unseen		SamPart3D	
	Suc↑	PGA↑	Suc↑	PGA↑	Suc↑	PGA↑
Wrap grasp	87.7	62.1	79.1	36.6	76.3	52.0
Pinch grasp	71.8	20.2	54.8	15.2	50.6	21.3

Table 3. **Ablation study for grasp style .**

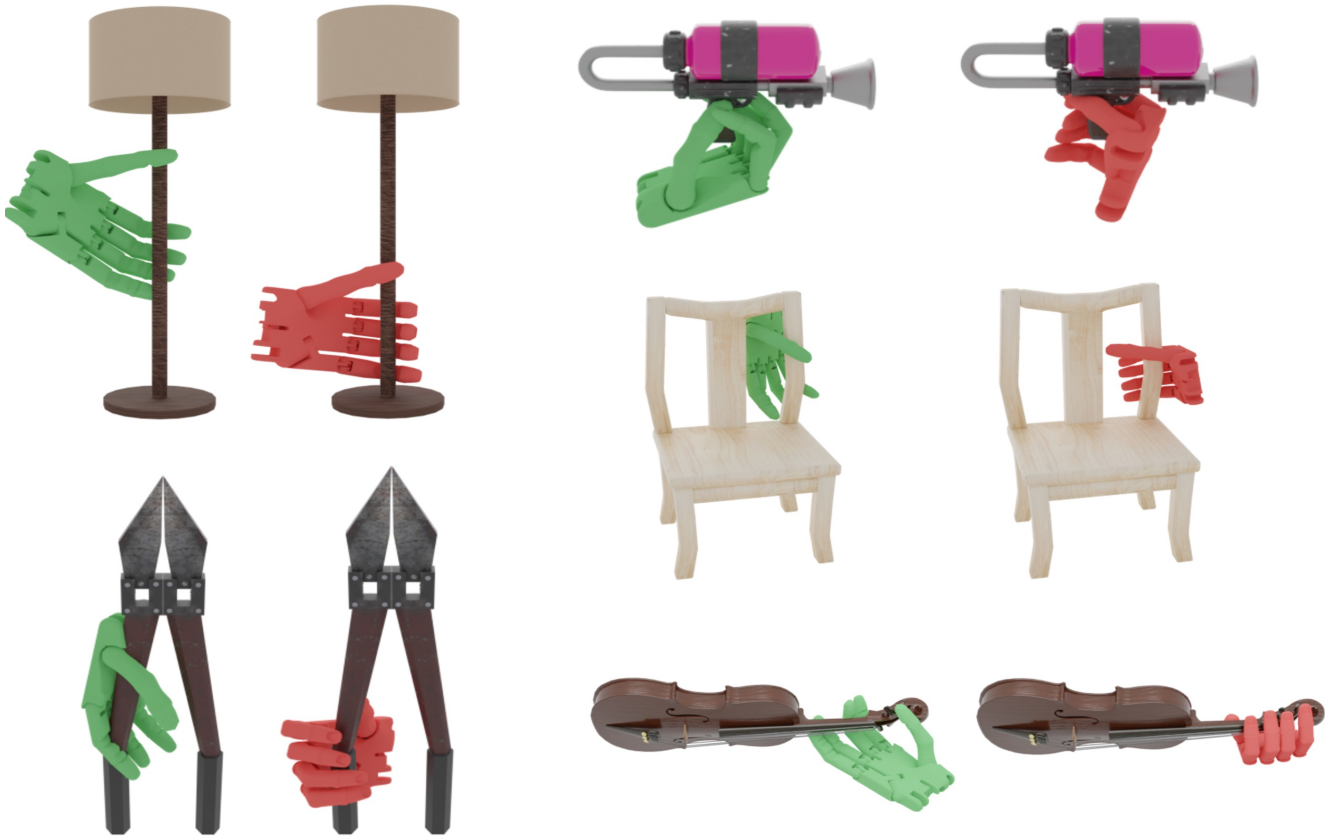


Figure 9. Comparison between grasp poses synthesized with random initialization colored in green v.s. our part-aligned initialization colored in red. The part-aligned initialization injects a strong prior about how a natural grasp is likely to be posed. The resulting optimized grasp poses with part-aligned initialization are more natural and semantically distinguishable, facilitating the instruction following training of VLG models.

Method	LVIS-Seen		Unseen		SamPart3D	
	Suc \uparrow	PGA \uparrow	Suc \uparrow	PGA \uparrow	Suc \uparrow	PGA \uparrow
DDPM	51.9	7.8	34.1	10.9	40.7	5.5
DDIM	57.7	12.5	39.6	10.4	35.2	8.5
FlowMatching	75.3	39.1	54.0	18.3	53.4	27.0

Table 4. Ablation study for denoising paradigms used in pose prediction.

Prompt used to query part name

[System Prompt]

You are a helpful AI assistant. The 6 multi-view images are from the same object (+x, -x, +y, -y, +z, -z). And the segmentation are labeled on the images as number marks. The object is from web and the user labels the object as [NAME] and describes it as [DESCRIPTION]. You need to

1. Count the number of visible labeled marks in images. You can provide a dictionary data structure called "image_info", which includes "part_num" key and "all_part_num". The value of "part_num" key is also a dictionary including image index, such as "image_0" and corresponding part num. The value of "all_part_num" is the total number of marked parts.
2. Recognize the object according to the images. You can give me a dictionary data structure called "object_info", which includes "object_name" key. The value of "object_name" is the name of the object.
3. Recognize the part name of each marked mask in each image. You can provide a dictionary called "part_info". The keys are part numbers, and the value of each key is the corresponding part name. For example, "part_info": {"part_0": "seat", "part_1": "backrest", "part_2": "leg", "part_3": "armrest", "part_4": "footrest"}.

Fine, after answering these questions, you give me dictionaries including "image_info", "object_info", and "part_info". Now you can give me a comprehensive dictionary called "gpt-4o-output", which includes these dictionaries.

Can you only output the comprehensive dictionary "gpt-4o-output" as json data format? The language should be English.

[IMAGE]



[Output]

```
{ "image_info": { "part_num": { "image_0": 3, "image_1": 4, "image_2": 5, "image_3": 5, "image_4": 3, "image_5": 3 }, "all_part_num": 5 }, "object_info": { "object_name": "DeWALT screwdriver" }, "part_info": { "0": "Chuck", "1": "Battery Pack", "2": "Handle", "3": "Motor Housing", "4": "Belt Clip" } }
```

Figure 10. Prompt used to query for part names of objects with GPT-4o. We use Set-of-Marks [23] with images rendered from six views. [NAME] and [DESCRIPTION] are the captions given in the OpenShape [13] dataset.

Prompt used to query object size

[System Prompt]

You are a helpful AI assistant. I will upload an image of the object. The object is from web and the BLIP algorithm gives a caption called [NAME] and user description is [DESCRIPTION]. You need to

1. Determine what kind of object it is. You can give me a dictionary data structure called "object_info", which includes "object_name" key. The value of "object_name" is the name of the object.
2. Determine the true size of the object. You can provide a dictionary data structure called "size_info", which includes keys of "length", "width", "height" and their corresponding values. The format is: "size_info": { "length": { "value": xxx, "unit": "cm" }, "width": { "value": xxx, "unit": "cm" }, "height": { "value": xxx, "unit": "cm" } }. Note that if you cannot determine the size, you can guess it as a reasonable value.

Fine, after answering these questions, you give me six dictionaries including "object_info", "size_info". Now you can give me a comprehensive dictionary called "gpt-4o-output", which includes these dictionaries.

Can you only output the comprehensive dictionary "gpt-4o-output" as json data format?

[IMAGE]



[Output]

```
{ "object_info": { "object_name": "Drill" }, "size_info": { "length": { "value": 20, "unit": "cm" }, "width": { "value": 7, "unit": "cm" }, "height": { "value": 23, "unit": "cm" } } }
```

Figure 11. Prompt used to query for object sizes with GPT-4o. [NAME] and [DESCRIPTION] are the captions given in the Open-Shape [13] dataset.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2
- [2] Hongkai Dai, Anirudha Majumdar, and Russ Tedrake. Synthesis and optimization of force closure grasps via sequential semidefinite programming. *Robotics Research: Volume 1*, pages 285–305, 2018. 1
- [3] Dawson-Haggerty et al. trimesh. 2
- [4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2
- [5] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020. 4
- [6] Carlo Ferrari, John F Canny, et al. Planning optimal grasps. In *ICRA*, page 6, 1992. 1
- [7] Álvaro González. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical geosciences*, 42:49–64, 2010. 4
- [8] Jiawei He, Lue Fan, Yuqi Wang, Yuntao Chen, Zehao Huang, Naiyan Wang, and Zhaoxiang Zhang. Tracking objects with 3d representation from videos. *arXiv preprint arXiv:2306.05416*, 2023. 4
- [9] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020. 2
- [10] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*, 2025. 4
- [11] Haosheng Li, Weixin Mao, Weipeng Deng, Chenyu Meng, Haoqiang Fan, Tiancai Wang, Ping Tan, Hongan Wang, and Xiaoming Deng. Multi-graspllm: A multimodal llm for multi-hand semantic guided grasp generation. *arXiv preprint arXiv:2412.08468*, 2024. 6
- [12] Kailin Li, Jingbo Wang, Lixin Yang, Cewu Lu, and Bo Dai. Semgrasp: Semantic grasp generation via language aligned discretization. In *European Conference on Computer Vision*, pages 109–127. Springer, 2025. 6
- [13] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in neural information processing systems*, 36:44860–44879, 2023. 8, 9
- [14] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator. *IEEE Robotics and Automation Letters*, 7(1): 470–477, 2021. 1
- [15] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, 2017. 1
- [16] Zekun Qi, Wenyao Zhang, Yufei Ding, Runpei Dong, Xinqiang Yu, Jingwen Li, Lingyun Xu, Baoyu Li, Xialin He, Guofan Fan, Jiazhaoh Zhang, Jiawei He, Jiayuan Gu, Xin Jin, Kaisheng Ma, Zhizheng Zhang, He Wang, and Li Yi. SoFar: Language-grounded orientation bridges spatial reasoning and object manipulation. *CoRR*, abs/2502.13143, 2025. 2, 4
- [17] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119. IEEE, 2023. 2
- [18] George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh: Zero-shot mesh part segmentation via lifting segment anything 2 to 3d. *arXiv preprint arXiv:2408.13679*, 2024. 2
- [19] Tokuo Tsuji, Kensuke Harada, and Kenji Kaneko. Easy and fast evaluation of grasp stability by using ellipsoidal approximation of friction cone. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1830–1837, 2009. 1
- [20] 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. 4, 6
- [21] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 2
- [22] 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. *arXiv preprint arXiv:2405.19291*, 2024. 6
- [23] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023. 8
- [24] Jialiang Zhang, Haoran Liu, Danshi Li, Xinqiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes. In *8th Annual Conference on Robot Learning*, 2024. 1, 6