

ThinkGrasp: A Vision-Language System for Strategic Part Grasping in Clutter

Anonymous Author(s)

Affiliation

Address

email

Abstract: Robotic grasping in cluttered environments remains a significant challenge due to occlusions and complex object arrangements. We have developed ThinkGrasp, a plug-and-play vision-language grasping system that makes use of GPT-4o’s advanced contextual reasoning for heavy clutter environment grasping strategies. ThinkGrasp can effectively identify and generate grasp poses for target objects, even when they are heavily obstructed or nearly invisible, by using goal-oriented language to guide the removal of obstructing objects. This approach progressively uncovers the target object and ultimately grasps it with a few steps and a high success rate. In both simulated and real experiments, ThinkGrasp achieved a high success rate and significantly outperformed state-of-the-art methods in heavily cluttered environments or with diverse unseen objects, demonstrating strong generalization capabilities.

Keywords: Robotic Grasping, Vision-Language Models, Language Conditioned Grasping

1 Introduction

The field of robotic grasping has seen significant advancements in recent years, with deep learning and vision-language models driving progress towards more intelligent and adaptable grasping systems [1, 2, 3]. However, robotic grasping in highly cluttered environments remains a major challenge, as target objects are often severely occluded or completely hidden [4, 5, 6]. Even state-of-the-art methods struggle to accurately identify and grasp objects in such scenarios.

To address this challenge, we propose ThinkGrasp, which combines the strength of large-scale pre-trained vision-language models with an occlusion handling system. ThinkGrasp leverages the advanced reasoning capabilities of models like GPT-4o [7] to gain a visual understanding of environment and object properties like sharpness, material, etc. By integrating this knowledge, ThinkGrasp can significantly improve success rates and ensure safer grasp poses by strategically removing occluding objects and focusing on the safest and most advantageous parts for grasping.

The main contributions of our work are as follows:

- We develop a plug-and-play occlusion handling system that efficiently utilizes visual and language information for robotic grasping. We further enhance the system’s reliability by implementing a robust error-handling framework that, based on LangSAM’s segmentation and scoring process utilizes iterative verification and corrective actions to address potential misidentifications by GPT-4o. These improvements significantly increase success rates, ensure safer grasp poses, and enhance reliability in diverse and cluttered environments.
- In the simulation, through extensive experiments on the challenging RefCOCO dataset [8], we demonstrate state-of-the-art performance. ThinkGrasp achieves a 98.0% success rate and fewer steps in cluttered scenes, outperforming prior methods like OVGNet [9] (43.8%) and VLG [10]

(75.3%). Despite unseen objects and heavy clutter levels, it maintains a high 78.9% success rate, showcasing its strong generalization capabilities. In the real world, ours achieved a high success rate with few steps.

- The modular design in our system enables easy adaptation to various robotic platforms and grasping systems. It demonstrates strong generalization capabilities, quickly adjusting to new language goals and novel objects by simple prompts, making it highly versatile and scalable.

2 Related Work

Robotic Grasping in Cluttered Environments: Robotic grasping in cluttered environments remains a significant challenge due to the complexity of occlusions and the diversity of objects. Traditional methods, which rely heavily on hand-crafted features and heuristics, struggle with generalization and robustness in diverse, unstructured environments [1, 11]. Deep learning methods that use CNNs and RL have shown improvements in grasp planning and execution [3, 12, 13, 14]. However, they often need a lot of data to be collected and labeled, which makes them less useful in a variety of situations [15, 4]. Recent methods like NG-Net [5] and Sim-Grasp [16] have made strides in cluttered environments but still face limitations in handling heavy clutter with diverse objects.

Pre-trained Models for Robotic Grasping: Vision-language models (VLMs) and large language models (LLMs) have shown promise in enhancing robotic grasping by integrating visual and language information[17]. Models such as CLIP [18] and CLIPort [19] have improved task performance, and VL-Grasp [20] has developed interactive grasp policies for cluttered scenes. Additionally, models like ManipVQA [21], RoboScript [22], CoPa [23], and OVAL-Prompt [24] use vision-language models and contextual information to improve the performance of grasping tasks. Voxposer [25] and GraspGPT [26] have demonstrated how LLMs can generate task-relevant actions and grasping strategies. Although these methods have advanced the field, they frequently have limitations due to their emphasis on particular types of clutter and the lack of a robust strategy for handling heavy occlusions.

3 Method

3.1 Problem Definition

In heavily cluttered environments, robotic grasping faces significant challenges due to occlusions and the presence of multiple objects. The main issue is coming up with an appropriate grasp pose for a target object that a natural language instruction specifies.

The key challenges are **1) Occlusions**, where objects are often partially or fully obscured by other items, making it difficult for the robot to identify and grasp the target object; **2) Ambiguity in Natural Language Instructions**, as instructions may be vague or ambiguous, requiring the robot to accurately interpret the user’s intent and identify the correct object among many possibilities; **3) Dynamic Environments**, where the grasping strategy needs to be changed in real-time as the positions and orientations of objects change; **4) Safety and Stability**, making sure that the grasp pose is not only possible but also safe and stable so that neither the objects nor the robot is damaged; and **5) Efficiency**, cutting down on the number of steps needed to grasp something so that the process is faster and more effective.

To solve these problems, we need a system that can correctly understand and interpret natural language instructions, find target objects even when they are partially obscured, change its grasp strategy based on the current environment, ensure safe and secure grasping, and work quickly so that tasks can be completed with less effort.

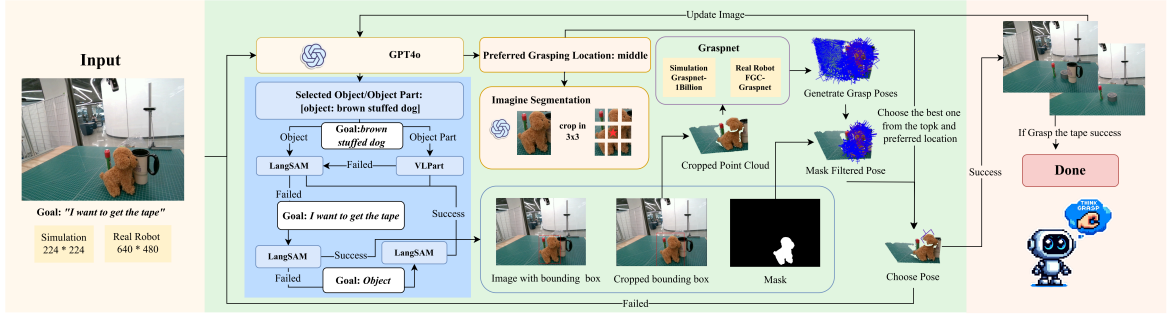


Figure 1: ThinkGrasp pipeline for cluttered environments

Our approach tackles the strategic part of grasping in cluttered environments via an iterative pipeline (Figure 1). Given an initial RGB-D scene observation O_0 (224×224 for simulation, 640×480 for real robot) and a natural language instruction g .

First, the system uses GPT-4o to do "imagine segmentation," which means it finds possible target objects or parts and suggests places to grab them within a 3×3 grid that is placed on top of the object's bounding box. Essentially, GPT-4o imagines what the segmented objects and ideal grasp locations could be based on the visual and language inputs. This involves identifying potential target objects and their parts that best match the instruction g and proposing candidate objects o_t and preferred grasping locations within a 3×3 grid for each object.

Combining GPT-4o's comprehension with a grid-based strategy enhances the precision of our system. It divides the cropping box containing the proposed target object or part into a 3×3 grid and suggests a number from 1 to 9 indicating the optimal grasping region (1 for top-left, 9 for bottom-right). This strategy, especially effective for low-resolution images, focuses on selecting optimal regions rather than exact points while also considering the constraints of the robotic arm and gripper for successful grasping.

Next, LangSAM[27] segments the image, focusing on the objects proposed by GPT-4o, and crops a point cloud containing these objects. The system iteratively refines its choices, using the cropped point cloud to update GPT-4o's "imagine segmentation," and predictions for the target object o_t and preferred grasping location. LangSAM and VLPART[28] handle object segmentation.

To determine the optimal grasping pose P_g , the system generates a set of candidate grasp poses A based on the cropped point cloud. In order to validate our system, we kept the variables consistent in our experiments. We used different grasp generation networks for simulation and real-world robot tests. Specifically, we employed Graspnet-1Billion [29] for all simulation comparisons while utilizing FGC-Graspnet [30] for real-robot comparisons. This approach ensures that our results are reliable and that any observed differences are due to the grasping system rather than inconsistencies in the grasp generation network. The candidate grasps A are evaluated based on their proximity to the preferred location suggested by GPT-4o and their grasp quality scores from the respective grasp generation module.

The system executes the optimal pose P_g for the selected target o_t . If the target is occluded or not visible, the system identifies and moves obstructing objects deemed most cost-effective to reveal the target.

This closed-loop process demonstrates the system's adaptability with the production of its next grasp strategy $P_{g,t+1}$ based on the updated scene observation O_{t+1} after each grasp attempt. The pipeline adjusts its grasping strategy as needed until the task is successfully completed or the maximum number of iterations is reached. It effectively manages the challenges presented by heavy clutter.

116 3.3 GPT-4o’s Role and Constraint Solver in Target Object Selection

117 Our grasping system leverages GPT-4o, a state-of-the-art vision-language model (VLM), to seam-
118 lessly integrate visual and language information. GPT-4o excels in contextual reasoning and knowl-
119 edge representation, making it particularly well-suited for complex grasping tasks in cluttered envi-
120 ronments.

121 **Target Object Selection:** GPT-4o identifies the object that best matches the provided instruction,
122 effectively locking it onto relevant regions. Unlike other models, GPT-4o avoids selecting irrelevant
123 objects, even without depth information. This capability ensures that the system does not attempt to
124 grasp objects that are unlikely to contain or conceal the goal object. For example, in Figure 2, the
125 small packet in the top left corner seems to have nothing hidden under it.

126 The target object selection process uses GPT-4o to choose the most relevant object based on the
127 given language instruction g and the scene context \mathbf{O}_t^c . This process considers factors such as object
128 relevance to the instruction, ease of grasping, and potential obstruction when making its selection.

129 The process can be formulated as:

$$o_t = \arg \max_o f_{\text{select}}(g, \mathbf{O}_t^c, o) \quad (1)$$

130 where o_t is the color and name of the selected target object, g is the language instruction, \mathbf{O}_t^c are
131 the color observations of the scene, and f_{select} represents the selection function that evaluates the
132 suitability of each object o in the context of the instruction and scene.

133 **Handling Occlusions and Clutter:** GPT-4o strategically identifies and selects objects, ensuring ac-
134 curate grasping even when objects are heavily occluded or partially visible. The system intelligently
135 removes occluding objects to improve visibility and grasp accuracy.

136 The appendix provides further technical details, including the structured process GPT-4o follows to
137 analyze and select the optimal grasp pose.

138 3.4 3×3 Grid Strategy for Optimal Grasp Part Selection

139 Our 3×3 grid strategy enhances the system’s ability to handle low-resolution images (224×224)
140 by shifting from selecting a precise point to choosing an optimal region within a 3×3 grid. This
141 transformation leverages broader contextual information, making the grasp selection process more
142 robust and reliable even with lower pixel density. The grid divides the target object, represented
143 by a bounding box that is derived from the highest-scoring output of the segmentation algorithm,
144 into nine cells. Each cell is evaluated based on safety, stability, and accessibility. GPT-4o outputs
145 a preferred grasping location within this grid, based on its imagined segmentation of the object,
146 guiding the subsequent segmentation and pose generation steps.

147 Unlike traditional methods that rely on a single best grasp pose selection, our system first evaluates
148 multiple potential grasp poses (top-k) based on their proximity to the preferred location. Then, from
149 these top candidates, the pose with the highest score is selected. This approach, combined with the
150 3×3 grid strategy to identify the optimal grasping region, ensures that the chosen grasp pose is both
151 optimal and stable, significantly enhancing overall performance and success rates.

152 3.5 Target Object Segmentation and Cropping Region Generation

153 **Segmentation and Cropping:** We use the LangSAM framework for generating precise segmenta-
154 tion masks and bounding boxes, which is good at segmenting low-resolution images. In cases where
155 GPT-4o identifies an object part, such as a handle, we utilize VLPART for fine-grained segmenta-
156 tion, ensuring detailed part identification. If VLPART fails, LangSAM combined with our 3×3 grid
157 strategy ensures robust performance.

158 **Grasp Pose Generation:** To determine the optimal grasping pose P_g , the system generates a set of
159 candidate grasp poses A based on the cropped point cloud. The candidate grasps A are evaluated

based on their proximity to the preferred location suggested by GPT-4o and their grasp quality scores from the respective grasp generation module. The grasp with the highest score after this evaluation is selected as the optimal grasp pose.

Robustness and Error Handling: Despite GPT-4o’s advanced capabilities, occasional misidentifications may occur. To mitigate this, we employ iterative verification and corrective actions, dynamically adjusting cropping regions and grasp strategies. This closed-loop control ensures continuous improvement based on real-time feedback, significantly enhancing robustness and reliability.

Our ablation experiments (Table 1) show that using LangSAM significantly improves system performance compared to using GPT-4o alone. By combining GPT-4o’s contextual understanding with LangSAM’s precise segmentation and VLPart’s fine-grained part identification, our system achieves higher success rates and efficiency metrics.

3.6 Grasp Pose Generation and Selection

Candidate Grasp Pose Generation: Using the local point cloud, the system generates a set of candidate grasp poses:

$$\mathbf{G} = f_{\text{grasp}}(\mathbf{P}_{\text{local}}) \quad (2)$$

where $\mathbf{P}_{\text{local}}$ represents the point cloud data within the cropped region.

Grasp Pose Evaluation: We use an analytic computation method to grade each grasp. Based on the improved force-closure metric from Graspnet-1Billion [29], the score is calculated by gradually decreasing the friction coefficient μ from 1 to 0.1 until the grasp is not antipodal. The lower the friction coefficient μ , the higher the probability of successful grasp. Our score s is defined as:

$$s = 1.1 - \mu$$

such that s lies in $(0, 1]$.

Each candidate grasp pose is evaluated based on its alignment with the preferred grasping location. The optimal grasp pose is selected by maximizing a score function that considers the suitability of each pose:

$$g_{\text{optimal}} = \arg \max_{g \in \mathbf{G}} \text{score}(g, p_{\text{preferred}})$$

Here, g_{optimal} is the optimal grasp pose, and $p_{\text{preferred}}$ is the preferred grasping location. The robot then performs the chosen ideal grasp pose (g_{ext} optimal).

3.7 Closed-Loop System for Robustness in Heavy Clutter

Our system enhances robustness in heavily cluttered environments through a closed-loop control mechanism that continuously updates the scene understanding after each grasp attempt, ensuring it works with the most current information. The cropping region and grasp poses are dynamically adjusted based on real-time feedback, allowing the system to focus on the most relevant areas and select the optimal grasp pose.

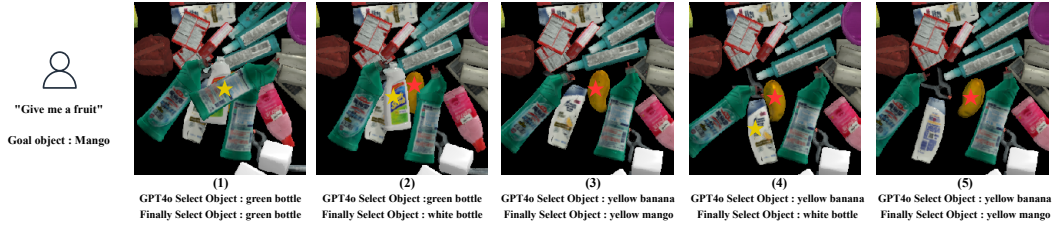


Figure 2: Closed-loop grasping process demonstrating

As shown in figure 2, the sequence of images demonstrates the process of selecting a target object based on a user’s command. First, the user provides the goal object "mango" and inputs the com-

mand "Give me a fruit". The initial color input image is from the simulation. GPT-4o selects an object (e.g., green bottle) and a preferred location based on the prompt, segmented into a 3×3 grid. This information is passed to LangSAM for segmentation. LangSAM segments all green bottles in the image and crops a point cloud that includes all the green bottles. It then generates all possible grasp poses within the cropped point cloud. The pose with the highest LangSAM segmentation score is selected as the target object. The target point is the center of the preferred object location provided by GPT-4o. From there, the system evaluates the top 10 poses closest to the target point and chooses the highest-scoring pose, which is then executed on the green bottle. Even if GPT-4o's initial selection doesn't match the goal (e.g., bottle instead of mango), LangSAM's segmentation and scoring process corrects errors and locks onto the intended target object due to distinct color features.

4 Experiments

Our system is designed to work effectively both in simulation and real-world settings, with tailored adaptations to address the unique challenges and constraints of each environment.

4.1 Simulation

Our simulation environment, built in PyBullet [31], involves a UR5 arm, a ROBOTIQ-85 gripper, and an Intel RealSense L515 camera. The raw images are resized to 224×224 pixels and segmented by LangSAM for precise object masks. We compare our solution against state-of-the-art methods, Vision-Language Grasping (VLG)[10] and OVGrasp[9], using the same GraspNet backbone for fair comparison. Additionally, we compare our method to directly use GPT-4o to select a target grasp point without additional processing or integration with other modules.



Figure 3: Clutter cases in simulation. The target objects are labeled with stars.



Figure 4: Heavy Clutter cases in simulation. The target objects are labeled with stars.

Our experiments focused on various tasks, such as grasping round objects, retrieving items for eating or drinking, and other specific requests. Each test case includes 15 runs, measured with two

metrics: Task Success Rate and Motion Number. The Task Success Rate is the average percentage of successful task completions within 15 action attempts over 15 test runs. Motion Number is the average number of motions per task completion.

Results. The results, summarized in Table 1, demonstrate that our system significantly outperforms the baselines in overall success rates and efficiency metrics. Specifically, our method achieved an average success rate of 0.980, with an average step count of 3.39 and an average success step count of 3.32 in clutter case (Figure 3). These results indicate that our system not only excels in accomplishing grasp tasks but also operates with greater efficiency, requiring fewer steps for successful task completion.

Table 1: Overall and Heavy Clutter Averages with Ablation Studies

Metric	VLG	OVGrasp	GPT4o (only)	no GPT4o	no 3×3	GPT crop	Ours
Overall Averages							
Average Success ↑	0.753	0.438	0.713	0.740	0.973	0.973	0.980
Average Step ↓	9.545	4.88	9.826	7.14	3.40	3.97	3.39
Average Success Step ↓	8.227	5.866	8.749	6.38	3.29	3.76	3.32
Heavy Clutter Overall Averages							
Average Success ↑	0.511	0.000	0.311	0.667	0.733	0.756	0.789
Average Step ↓	32.98	NA	40.25	22.04	18.71	20.48	19.35
Average Success Step ↓	25.27	NA	33.48	20.50	16.50	16.89	17.06

We also evaluated our system’s performance in heavy clutter scenarios, where objects are partially or completely occluded. These scenarios (Figure 4) involve up to 30 unseen objects and allow up to 50 action attempts per run. The results, shown in Table 1, demonstrate that our system significantly outperforms the baselines in these challenging conditions, achieving the highest success rates and the fewest steps required for successful grasps.

Table 2: Heavy Clutter Average Success ↑

Task	VLG	OVGrasp	GPT4o(only)	Ours
grasp a ball	0.467	0.000	0.800	1.000
grasp a ball (CI)	0.867	0.000	0.400	0.933
get something to hold other things	0.067	0.000	0.000	0.133
get something to hold other things (CI)	0.400	0.000	0.533	0.800
I need a fruit	0.467	0.000	0.133	0.867
I need a fruit (CI)	0.800	0.000	0.000	1.000

Ablation study. To assess the contribution of different components of our system, we conducted ablation studies. The results of these ablation studies, shown in Table 1, highlight the effectiveness of our complete system. The ”no 3×3” configuration refers to an approach where the system does not divide the object into a 3×3 grid to select the grasp point but instead uses a fixed or pre-determined position. The ”GPT crop” configuration uses GPT-4o to output crop coordinates for the point cloud, focusing on the relevant area for grasping. The ”no GPT4o” configuration removes the use of GPT-4o. These experiments demonstrate that our complete system, which integrates all these components, achieves superior performance.

4.2 Real-World Experiments

We extended our system’s capabilities to real-world environments to handle complex and variable scenarios. The setup included a UR5 robotic arm with 6 DoFs and a Robotiq 85 gripper. Observations were captured using a RealSense D455 camera, providing both color and depth images for point cloud construction. The target pose for grasping was determined using the MoveIt motion planning framework with the RRT* algorithm. ROS managed the communication, running on a

workstation equipped with a 12GB 2080Ti GPU. Our ThinkGrasp model, deployed using Flask on a server with dual 3090 GPUs, provided grasp pose predictions within 10 seconds via the GPT-4o API.

In our real-world experiments⁵ we compared our system against VL-Grasp, using the same FGC-GraspNet downstream grasp model to ensure a fair assessment of the improvements introduced by our strategic part grasping and heavy clutter handling mechanisms.

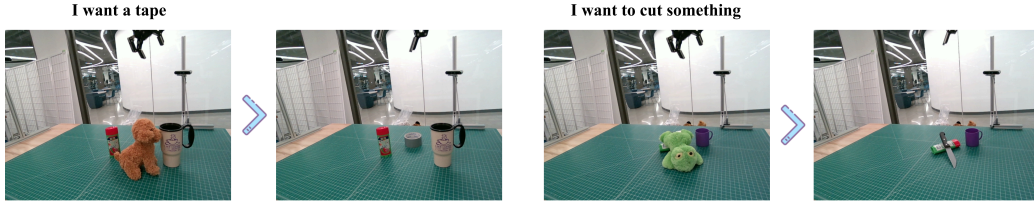


Figure 5: Real Robot Task

Table 3: Real-World Clutter Task Performance

Clutter Task	Step 1 Success Rate	Step 2 Success Rate
I want a tape	15/20 (75%) success to get the toy dog	12/15 (80%) grasp tape
I want to cut something	18/20 (90%) success to get the toy frog	10/18 (55.6%) grasp knife by handle

Results. Our results ^{3,11} show that our system has a high success rate in identifying and grasping target objects, even in cluttered environments. The use of VLPART and GPT-4o significantly improved robustness and accuracy. However, failures were sometimes due to limitations in single image data, suboptimal grasp poses from the downstream model, and variations in the UR5 robot’s stability and control. These findings underline the importance of robust image processing, high-quality grasp pose generation, and stable robotic control. Further technical details and experimental setups are provided in the appendix (Table A).

5 Conclusion

This paper presents a novel plug-and-play vision-language behavior modeling approach for robotic grasping in cluttered environments. By leveraging GPT-4o’s advanced contextual reasoning and VLPART’s precise segmentation, our system effectively identifies and grasps target objects, even when they are heavily occluded. Through extensive simulation and real-world experiments, our approach demonstrated superior performance and robustness compared to existing methods.

However, our approach has some limitations. It is currently designed to perform grasp tasks only, and the grasp poses generated may suffer from occlusions or inaccuracies due to the single-view point cloud reconstruction, potentially causing collisions or incomplete grasps. Future work will address these limitations by incorporating multi-view point cloud integration and expanding the range of tasks beyond grasping.

References

- [1] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.
- [2] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- [3] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.
- [4] U. Viereck, A. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Conference on robot learning*, pages 291–300. PMLR, 2017.
- [5] M. Shi, J. Hou, Z. Li, and D. Zhu. Ng-net: No-grasp annotation grasp detection network for stacked scenes. *Journal of Intelligent Manufacturing*, pages 1–14, 2024.
- [6] Y. Liu, A. Qualmann, Z. Yu, M. Gabriel, P. Schillinger, M. Spies, N. A. Vien, and A. Geiger. Efficient end-to-end detection of 6-dof grasps for robotic bin picking. *arXiv preprint arXiv:2405.06336*, 2024.
- [7] OpenAI. Gpt-4o: Openai’s multimodal vision-language system. <https://openai.com/research/gpt-4o>, 2023. Accessed: 2024-06-05.
- [8] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 69–85. Springer, 2016.
- [9] M. Li, Q. Zhao, S. Lyu, C. Wang, Y. Ma, G. Cheng, and C. Yang. Ovgnet: An unified visual-linguistic framework for open-vocabulary robotic grasping. <https://github.com/cv516Buaa/OVGNet>, 2024. Accessed: 2024-05-02.
- [10] K. Xu, S. Zhao, Z. Zhou, Z. Li, H. Pi, Y. Zhu, Y. Wang, and R. Xiong. A joint modeling of vision-language-action for target-oriented grasping in clutter. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11597–11604. IEEE, 2023.
- [11] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.
- [12] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.
- [13] H. Ma, R. Qin, B. Gao, D. Huang, et al. Sim-to-real grasp detection with global-to-local rgb-d adaptation. *arXiv preprint arXiv:2403.11511*, 2024.
- [14] S. Chen, R. Garcia, I. Laptev, and C. Schmid. Sugar: Pre-training 3d visual representations for robotics. *arXiv preprint arXiv:2404.01491*, 2024.
- [15] J. Mahler and K. Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524. PMLR, 2017.
- [16] J. Li and D. J. Cappelleri. Sim-grasp: Learning 6-dof grasp policies for cluttered environments using a synthetic benchmark. *arXiv preprint arXiv:2405.00841*, 2024.
- [17] Y. Zhang, Z. Ma, X. Gao, S. Shakhia, Q. Gao, and J. Chai. Groundhog: Grounding large language models to holistic segmentation, 2024.

- [18] 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*, pages 8748–8763. PMLR, 2021.
- [19] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.
- [20] Y. Lu, Y. Fan, B. Deng, F. Liu, Y. Li, and S. Wang. Vl-grasp: a 6-dof interactive grasp policy for language-oriented objects in cluttered indoor scenes. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 976–983. IEEE, 2023.
- [21] S. Huang, I. Ponomarenko, Z. Jiang, X. Li, X. Hu, P. Gao, H. Li, and H. Dong. Manipvqa: Injecting robotic affordance and physically grounded information into multi-modal large language models. *arXiv preprint arXiv:2403.11289*, 2024.
- [22] J. Chen, Y. Mu, Q. Yu, T. Wei, S. Wu, Z. Yuan, Z. Liang, C. Yang, K. Zhang, W. Shao, et al. Roboscript: Code generation for free-form manipulation tasks across real and simulation. *arXiv preprint arXiv:2402.14623*, 2024.
- [23] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao. Copa: General robotic manipulation through spatial constraints of parts with foundation models. *arXiv preprint arXiv:2403.08248*, 2024.
- [24] E. Tong, A. Opipari, S. Lewis, Z. Zeng, and O. C. Jenkins. Oval-prompt: Open-vocabulary affordance localization for robot manipulation through llm affordance-grounding. *arXiv preprint arXiv:2404.11000*, 2024.
- [25] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.
- [26] C. Tang, D. Huang, W. Ge, W. Liu, and H. Zhang. Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping. *IEEE Robotics and Automation Letters*, 8(11):7551–7558, 2023. doi:10.1109/LRA.2023.3320012.
- [27] L. Medeiros. Language segment-anything. <https://github.com/luca-medeiros/lang-segment-anything>.
- [28] P. Sun, S. Chen, C. Zhu, F. Xiao, P. Luo, S. Xie, and Z. Yan. Going denser with open-vocabulary part segmentation. *arXiv preprint arXiv:2305.11173*, 2023.
- [29] H.-S. Fang, C. Wang, M. Gou, and C. 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.
- [30] Y. Lu, B. Deng, Z. Wang, P. Zhi, Y. Li, and S. Wang. Hybrid physical metric for 6-dof grasp pose detection. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8238–8244. IEEE, 2022.
- [31] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

347 A Appendix

348 A.1 Prompt

Algorithm 1 Prompt

- 1: **Given a 224×224 input image and the provided instruction, perform the following steps:**
 - 2: **Target Object Selection:**
 - 3: Identify the object in the image that best matches the instruction. If the target object is found, select it as the target object.
 - 4: If the target object is not visible, select the most cost-effective object or object part considering ease of grasping, importance, and safety.
 - 5: If the object has a handle or a part that is easier or safer to grasp, select the part. [for example the handle of a knife]
 - 6: Consider the geometric shape of the objects and the gripper's success rate when selecting the target object or object part.
 - 7: Output the name of the selected object or object part as [object:color and name] or [object part:color and name].
 - 8: Round object means like ball. Cup is different from mug.
 - 9: **Cropping Box Calculation:**
 - 10: Calculate a cropping box that includes the target object and all surrounding objects that might be relevant for grasping.
 - 11: Provide the coordinates of the cropping box in the format (top-left x, top-left y, bottom-right x, bottom-right y).
 - 12: **Object Properties within Cropping Box:**
 - 13: For each object within the cropping box, provide the following properties:
 - 14: Grasping Score: Evaluate the ease or difficulty of grasping the object on a scale from 0 to 100 (0 being extremely difficult, 100 being extremely easy).
 - 15: Preferred Grasping Location: Divide the cropping box into a 3×3 grid and return a number from 1 to 9 indicating the preferred grasping location (1 for top-left, 9 for bottom-right).
 - 16: Additionally, consider the preferred grasping location that is most successful for the UR5 robotic arm and gripper.
 - 17: **Output should be in the following format:**
 - 18: Selected Object/Object Part: [object:color and name] or [object part:color and name]
 - 19: Cropping Box Coordinates: (top-left x, top-left y, bottom-right x, bottom-right y)
 - 20: Objects and Their Properties:
 - 21: Object: [color and name]
 - 22: Grasping Score: [value]
 - 23: Preferred Grasping Location: [value]
 - 24: **Example Output:**
 - 25: Selected Object/Object Part: [object:blue ball]
 - 26: Cropping Box Coordinates: (50, 50, 200, 200)
 - 27: Objects and Their Properties:
 - 28: Object: Blue Ball
 - 29: Grasping Score: 90
 - 30: Preferred Grasping Location: middle
 - 31: Object: Yellow Bottle
 - 32: Grasping Score: 75
 - 33: Preferred Grasping Location: top-right
-

349 A.2 Detailed Process of GPT-4o and Constraint Solver

350 **Cropping Box Calculation:** GPT-4o calculates a cropping box that includes the target object and
351 relevant surrounding objects, ensuring focused and effective grasping.

352 **Object Properties within Cropping Box:** GPT-4o assesses the grasping difficulty for each object
353 within the cropping box and identifies the optimal grasp location within a 3×3 grid. This detailed
354 analysis ensures the selection of the safest and most practical grasp points.

By integrating these steps, GPT-4o ensures the selected grasp pose is feasible and optimal, considering all relevant factors. This method leverages GPT-4o’s advanced understanding to interpret complex instructions and make informed decisions, significantly enhancing robustness and success rates in cluttered environments.

A.3 Results

Tables 4, 5, 6, 7, 8, 9, 10, and 11 present the performance of our approach compared to baseline methods across various tasks. Our method consistently achieves high success rates and lower average steps, demonstrating robustness and efficiency. Notably, in tasks such as "Get something to eat" and "Give me the cup," our system outperforms other methods, indicating its ability to identify and grasp target objects even in cluttered environments accurately. However, the heavy clutter scenarios highlight limitations, such as increased average steps due to the single-view point cloud reconstruction, which can lead to potential collisions or incomplete grasps.

Table 4: Average Success \uparrow

Task	VLG	OVGrasp	GPT4o(only)	Ours
Grasp a round object	0.933	1.000	1.000	1.000
Get something to eat	1.000	0.000	0.800	1.000
Get something to hold other things	0.933	0.000	0.600	1.000
I want a round object	1.000	1.000	0.533	0.867
Give me the cup	0.800	0.000	0.333	0.933
I need a cup	1.000	0.375	0.800	1.000
I need a fruit	0.733	1.000	0.933	1.000
Get something to drink	0.133	0.000	0.467	1.000
Give me the theramed	0.200	0.000	0.667	1.000
Give me the pear	0.800	1.000	1.000	1.000

366

Table 5: Average Step \downarrow

Task	VLG	OVGrasp	GPT4o(only)	Ours
Grasp a round object	6.47	8.00	5.40	4.40
Get something to eat	4.20	NA	7.80	2.00
Get something to hold other things	9.60	NA	13.33	2.27
I want a round object	8.47	2.00	12.93	7.07
Give me the cup	9.93	NA	14.53	6.20
I need a cup	10.31	4.40	8.80	3.93
I need a fruit	8.67	2.00	5.40	3.13
Get something to drink	14.47	NA	10.13	1.67
Give me the theramed	14.20	NA	12.07	2.00
Give me the pear	9.13	6.00	3.87	1.27

Table 6: Average Success Step ↓

Task	VLG	OVGrasp	GPT4o(only)	Ours
Grasp a round object	5.86	8.00	5.40	<u>4.40</u>
Get something to eat	4.20	NA	6.00	<u>2.00</u>
Get something to hold other things	9.21	NA	12.22	<u>2.27</u>
I want a round object	8.47	<u>2.00</u>	11.13	6.00
Give me the cup	8.67	NA	13.60	<u>5.57</u>
I need a cup	9.33	9.33	7.25	<u>3.93</u>
I need a fruit	6.36	<u>2.00</u>	5.71	3.13
Get something to drink	12.50	NA	7.71	<u>1.67</u>
Give me the theramed	11.00	NA	10.60	<u>2.00</u>
Give me the pear	7.67	6.00	3.87	<u>1.27</u>

Table 7: Heavy Clutter Average Step ↓

Task	VLG	OVGrasp	GPT4o(only)	Ours
grasp a ball	25.40	NA	39.33	<u>19.20</u>
grasp a ball (CI)	25.40	NA	43.73	<u>21.33</u>
get something to hold other things	34.53	NA	NA	<u>14.27</u>
get something to hold other things (CI)	28.60	NA	31.53	<u>17.53</u>
I need a fruit	46.07	NA	48.40	<u>25.87</u>
I need a fruit (CI)	35.87	NA	NA	<u>19.93</u>

Table 8: Heavy Clutter Average Success Step ↓

Task	VLG	OVGrasp	GPT4o(only)	Ours
grasp a ball	21.61	NA	34.33	<u>19.20</u>
grasp a ball (CI)	21.61	NA	34.33	<u>19.28</u>
get something to hold other things	12.00	NA	NA	<u>5.00</u>
get something to hold other things (CI)	26.50	NA	27.25	<u>17.25</u>
I need a fruit	41.57	NA	38.00	<u>22.69</u>
I need a fruit (CI)	32.33	NA	NA	<u>19.93</u>

Table 9: Case Comparisons

Case	Method	avg_success \uparrow	avg_step \downarrow	avg_success_step \downarrow
Grasp a round object	no 3 \times 3	<u>1.00</u>	4.27	4.27
	no GPT4o	<u>1.00</u>	6.87	6.87
	GPT crop	<u>1.00</u>	<u>3.47</u>	<u>3.47</u>
	Ours	<u>1.00</u>	4.40	4.40
Get something to eat	no 3 \times 3	<u>1.00</u>	2.27	2.27
	no GPT4o	<u>1.00</u>	2.87	2.87
	GPT crop	<u>1.00</u>	2.33	2.33
	Ours	<u>1.00</u>	<u>2.00</u>	<u>2.00</u>
Get something to hold other things	no 3 \times 3	<u>1.00</u>	<u>2.20</u>	<u>2.20</u>
	no GPT4o	0.40	14.00	12.50
	GPT crop	0.933	9.00	8.57
	Ours	<u>1.00</u>	2.27	2.27
I want a round object	no 3 \times 3	<u>0.933</u>	<u>5.80</u>	<u>5.14</u>
	no GPT4o	0.600	10.27	7.67
	GPT crop	0.800	5.93	4.25
	Ours	0.867	7.07	6.00
Give me the cup	no 3 \times 3	<u>1.00</u>	<u>6.20</u>	6.20
	no GPT4o	0.800	6.67	6.25
	GPT crop	<u>1.00</u>	5.40	5.40
	Ours	0.933	<u>6.20</u>	<u>5.57</u>
I need a cup	no 3 \times 3	0.867	4.07	3.54
	no GPT4o	0.533	12.13	9.63
	GPT crop	<u>1.00</u>	<u>2.53</u>	<u>2.53</u>
	Ours	<u>1.00</u>	3.93	3.93
I need a fruit	no 3 \times 3	<u>1.00</u>	3.20	3.20
	no GPT4o	0.733	11.00	9.55
	GPT crop	<u>1.00</u>	3.87	3.87
	Ours	<u>1.00</u>	<u>3.13</u>	<u>3.13</u>
Get something to drink	no 3 \times 3	0.933	<u>1.53</u>	<u>1.57</u>
	no GPT4o	0.400	12.13	10.00
	GPT crop	<u>1.00</u>	2.47	2.47
	Ours	<u>1.00</u>	1.67	1.67
Give me the theramed	no 3 \times 3	<u>1.00</u>	<u>1.87</u>	<u>1.87</u>
	no GPT4o	<u>1.00</u>	2.07	2.07
	GPT crop	<u>1.00</u>	2.47	2.47
	Ours	<u>1.00</u>	2.00	2.00
Give me the pear	no 3 \times 3	<u>1.00</u>	1.60	1.60
	no GPT4o	0.933	1.40	1.43
	GPT crop	<u>1.00</u>	<u>1.27</u>	<u>1.27</u>
	Ours	<u>1.00</u>	<u>1.27</u>	<u>1.27</u>

Table 10: Heavy Clutter Case Comparisons

Case	Method	avg_success \uparrow	avg_step \downarrow	avg_success_step \downarrow
Grasp a ball	no 3 \times 3	0.933	22.33	20.36
	no GPT4o	0.667	28.60	29.60
	GPT crop	0.933	25.87	24.14
	Ours	<u>1.000</u>	<u>19.20</u>	<u>19.20</u>
Grasp a ball [CI]	no 3 \times 3	<u>1.000</u>	20.27	20.27
	no GPT4o	<u>1.000</u>	<u>12.47</u>	<u>12.47</u>
	GPT crop	0.933	19.00	16.79
	Ours	0.933	21.33	21.33
Get something to hold other things	no 3 \times 3	0.067	11.47	6.00
	no GPT4o	<u>0.200</u>	<u>7.87</u>	<u>4.00</u>
	GPT crop	0.067	15.73	4.00
	Ours	0.133	14.27	5.00
Get something to hold other things [CI]	no 3 \times 3	0.467	16.07	12.14
	no GPT4o	0.533	<u>11.87</u>	12.38
	GPT crop	<u>0.800</u>	15.93	15.83
	Ours	<u>0.800</u>	17.53	<u>17.25</u>
I need a fruit	no 3 \times 3	<u>0.933</u>	<u>23.47</u>	<u>21.57</u>
	no GPT4o	0.733	38.27	34.00
	GPT crop	0.800	27.07	21.33
	Ours	0.867	25.87	22.69
I need a fruit [CI]	no 3 \times 3	<u>1.000</u>	18.67	18.67
	no GPT4o	0.867	33.13	30.54
	GPT crop	<u>1.000</u>	<u>19.27</u>	<u>19.27</u>
	Ours	<u>1.000</u>	19.93	19.93

Table 11: VL-Grasp Real-World Clutter Task Performance

Task Step	I want a tape	I want to cut something
Step 1 Success Rate	11/20 (55%) success to get the toy dog	9/20 (45%) success to get the toy frog
Step 2 Success Rate	0/11 grasp tape, 6/11 (54.5%) success to get the red and green object	2/9 (22.2%) grasp knife by handle