

# Towards Open-World Grasping with Large Vision-Language Models

## Supplementary Material

### A LMM prompts and response examples

#### A.1 Prompts

We present the text prompts for all use cases considered in this work, namely: referring segmentation (Fig. 1), grounded grasp planning (Fig. 2) and grasp ranking (Fig. 3). Fields in purple correspond to variable input; either the grasping instruction in the referring segmentation prompt (i.e., `user_input`) or the label ID of the target object to grasp for the planning prompt, as predicted by the LMM grounder (i.e., `target`). The visual prompts are the marked images as explained in the methodology section of our paper and more examples are illustrated in the next subsection. The system message is defined always as: “*For any marker IDs mentioned in your answer, please highlight them with []*.”

In this work we utilized the GPT-4v(ision) model [1] via its alpha release in the OpenAI web API. In the following, we summarize our key findings while experimenting with multimodal prompts for GPT-4v.

**Clarity of visual markers** We find that the most common failure mode of visual marker prompting with GPT-4v is that it sometimes struggles to discriminate which ID corresponds to what segment. Especially in cluttered scenes, label IDs might severely overlap within small frame regions. Several techniques can assist in making the markers more clear to the VLM. We adopt the algorithm of [2] for overlaying numeric IDs within the frame with minimal overlap. Further, we found that coloring both the internal of each segment’s mask and its ID with the same unique color also helps in better VLM interpretation. Colors are chosen to be visually distinguishable. Finally, increasing the resolution of the marked image and the size layout of the markers also proves useful.

**Reference Image** If not highlighting the internal of each segment, we find that GPT-4v sometimes refers to regions with wrong IDs, especially in highly cluttered scenes. But if the masks are highlighted with high opacity, then the appearance of the object becomes less visible and GPT-4v struggles to recognize it. We propose a technique to ameliorate this is by passing both the original (reference) and the marked image and constructing a text prompt that explains that the latter corresponds to annotated segments of the first.

**Chain-of-thoughts** Chain-of-Thought (CoT) prompting is a well-established methodology for guiding LLMs to perform multi-step reasoning and reduce hallucinations. We find that VLMs share

You are highly skilled in grounding natural language descriptions to matching objects in an image. You are given two images: the raw image and a marked image. The marked image is an exact replica of the raw image, but each object is highlighted with a unique color and a numeric ID. The color of each object's highlighted mask corresponds to the color of the unique ID, so you can determine which object corresponds to what ID. If the given natural language description contains spatial relations (e.g. 'bowl left from mug'), you should assume the perspective of the viewer in order to resolve the spatial relations. If the given description contains colors (e.g. 'the red soda can'), then we mean the actual color of the object in the raw image, not to be confused with the highlighted color in the marked image. If you find the target object in the raw image, then you should look for its ID in the marked image. It is crucial to remember that the object will be in the same place in the marked as in the raw image. Please reach your answer by thinking step-by-step. Always finish your response with: 'My final answer is: [the label ID]'.

Provide the label ID that best matches the description: '`user_input`'.

Figure 1: Referring Segmentation Text Prompt

You are highly skilled in robotic task planning, able to determine a plan to ensure a target object is graspable. You will be given an instruction, which refers to the target object to grasp. If the object is in sight, you need to directly grasp it. If the target object is blocked by other objects, you need to remove all the blocking objects before picking up the target object.

You have to respond with a numbered list of objects to manipulate, by referring to their numeric IDs. Here is an example:

1. remove [ID]
2. remove [ID]
3. pick [ID]

It's essential to stick to the above format. When creating a plan, replace the placeholders [IDs] with numeric IDs of specific objects in the image. Before you create the plan, please write a small paragraph where you explain what objects are blocking the target object, by referring to their numeric IDs.

Task instruction: "Grasp object `target`".

Figure 2: Grounded Grasp Planning Text Prompt

similar properties and prompting them to reason about their final answer before producing it can robustify the response quality. For grounding, we ask GPT-4v to decompose the input instruction in steps and refer to all intermediate referenced objects. For grasp planning, we ask it to explicitly mention all objects that are blocking the target object by their numeric ID, before producing a plan. For grasp ranking, we decompose the prompt in three steps: (i) identify the category of the target object and provide a general description of what constitutes a good grasp for it given its shape, (ii) list the grasp IDs that will most likely lead to contact with neighboring objects, and (iii) rank the grasp IDs based on the previous two steps.

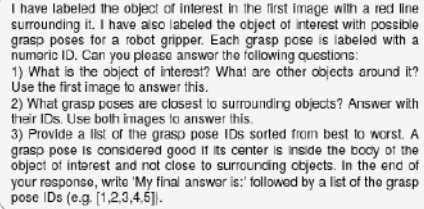
**Self-consistency** Even with zero temperature, we observe that the outputs of GPT-4v are not always reproducible. The reason for this is unknown since the actual model specifics are unknown, but a popular speculation is that GPT-4 is a *Mixture-of-Expert*-based model [3] that has implicit stochasticity. We find that sometimes GPT-4v might produce different responses at different runs, even with exactly the same prompt. In an attempt to reduce the effect of this phenomenon and robustify VLM outputs, we use the self-consistency method developed for LLMs [4].

In particular, we ask GPT-4v to provide multiple responses, parse each one separately and then perform majority voting to determine the most consistent output.

**In-context examples** In-context examples have shown to be a powerful asset for allowing LLMs to learn from a few examples [5]. In our study, we find that VLMs have similar capabilities, which can provide improvements in the grasp planning and contact reasoning stages through in-context examples. Please see Appendix E for further discussion.

**In-context examples** In-context examples have shown to be a powerful asset for allowing LLMs to learn from a few examples [5]. In our study, we find that VLMs have similar capabilities, which can improve the robustness of the grasp planning and contact reasoning stages. Both the image and an exemplar response are appended to the prompt to GPT-4v, emulating conversation history as in a chatbot setting. Similar to [6], we find that providing both a positive and a negative example (e.g. for grasp planning, when to pick the target directly and when to attempt to declutter first) can aid in enabling the VLM to understand the desired behavior

**Structured outputs** In order to ensure that the output of the VLM is parsable, we ask it to wrap its final answer with a standard format. We find that simply mentioning the desired response format in the input prompt is sufficient to produce parsable outputs, similar to previous instruct-tuned LLMs [7].



I have labeled the object of interest in the first image with a red line surrounding it. I have also labeled the object of interest with possible grasp poses for a robot gripper. Each grasp pose is labeled with a numeric ID. Can you please answer the following questions:

- 1) What is the object of interest? What are other objects around it? Use the first image to answer this.
- 2) What grasp poses are closest to surrounding objects? Answer with their IDs. Use both images to answer this.
- 3) Provide a list of the grasp pose IDs sorted from best to worst. A grasp pose is considered good if its center is inside the body of the object of interest and not close to surrounding objects. In the end of your response, write 'My final answer is:' followed by a list of the grasp pose IDs (e.g. [1,2,3,4,5]).

Figure 3: Grasp Ranking Text Prompt

## A.2 Example Responses

In Figs. 7, 8, 9, we provide example responses for grounding different types of language queries in OCID scenes. We observed that GPT-4v, augmented with marked image prompting, can ground not just object-related queries but also complex referring expressions that require reasoning about space, visual attributes, semantics and user-affordances. Interestingly, we find that GPT-4v responds to queries that require symbolic reasoning concepts such as counting and negation, which are notoriously hard to emerge in specialist grounding models. In Fig. 10, we provide some example responses corresponding to failure cases. Main failure modes include: a) grounding a distractor instead of the desired object, b) not finding the object of interest at all, c) providing a correct reasoning and identifying the target in the raw image, but providing a wrong ID of an irrelevant object.

Name	Attribute	Spatial Rel.	Visual Rel.	Sem. Rel.	Multi-hop	Affordance	Total
42	26	33	19	13	24	16	173

Table 1: Number of samples in grounding evaluation dataset.

## B Robot experiments

### B.1 Setups

Our object catalog for seen/unseen trials is shown in Fig. 4. In Gazebo, isolated scenarios are generated by ensuring all spawned objects have a fixed 3D distance, while in cluttered scenarios we ensure contact between the target object and neighbouring objects, by first spawning the target and then sampling different poses for other object models around it. In real-robot experiments, we manually setup the scenes while making sure to replicate the setup exactly for fair comparisons between baselines. In all trial scenes that contain distractor objects, the user instruction refers to some property that disambiguates the target instance from other objects of the same category, using names, attributes and spatial relations. We also conduct experiments without distractors for affordance-based queries, which require semantic reasoning to be correctly grounded.

We use the default `torchvision` implementation of Mask-RCNN, with the model weights provided by PyTorch Hub. We visually inspect the segmentation masks and determined that their output is sufficient both for synthetic and real-world images. For grasp synthesis, we generate a top-down orthographic projection of the scene, both for color and for depth (i.e. reverse depth - heightmap). This is the input we pass to the pretrained GR-ConvNet. In order to align regions from the 2D frame where Mask-RCNN provides segmentations and the orthographic projection where our grasp synthesis model provides grasp poses, we use the Hungarian matching algorithm to match the centers of outputs from both models, after projected to 3D and transformed to a world reference frame (robot base). We use the 3D euclidean distance between regions as the cost function for the algorithm.



Figure 4: Seen (*left column*) and unseen (*right column*) object used in our robot experiments in Gazebo (*top*) and the real world (*bottom*).

### B.2 Baseline Implementation

**CROG** CROG receives an single  $448 \times 448$  RGB view and a natural language query, and provides both an instance segmentation mask for the target object, as well as a set of 4-DoF grasp proposals, assuming that the gripper approaches the object aligned with the perspective of the camera. We use the checkpoint provided by the original paper, trained in the multiple split of OCID-VLG dataset, which contains 90k scene-query-grasp data from around 1,000 unique scenes from 31 object categories. The model uses CLIP’s pretrained ResNet-50 visual and BERT text encoders, but fine-tunes them end-to-end in OCID scenes for joint grounding and grasp synthesis tasks.

**SayCan-IM** We build an LLM-based baseline that follows the general architecture of SayCan, i.e. generating the next action that the robot should take in an autoregressive fashion. We use the gpt-4-turbo LLM engine for plan generation and prompt it with few-shot examples that follow the reason-then-act format, as introduced in InnerMonologue, and further optimized in the ReACT work. The primitive action library is identical to our OWG implementation, i.e. two primitive actions: `remove` and `pick`, corresponding to picking and placing a blocking object in a pre-defined region, and grasping the object of interest respectively. The prompt contains two in-context examples, which demonstrate when to select to remove the closest object(s), based on the relative distance of the

Method	Found. Model	Name	Attribute	Spatial Relation	Visual Relation	Semantic Relation	Affordance	Multi-hop	Avg.
PolyFormer	-	20.9	13.3	2.6	0.8	3.1	6.7	8.3	8.0
SEEM	-	23.3	10.1	4.6	10.5	10.2	7.9	17.5	12.1
ReCLIP	CLIP	36.9	40.0	12.7	14.2	20.1	23.0	34.0	25.9
RedCircle	CLIP	33.3	21.1	19.7	15.4	18.8	24.0	47.4	25.7
FDVP	CLIP	25.1	19.0	23.7	25.2	12.3	22.5	22.8	21.6
SoM	GPT-4v	40.1	25.0	23.3	40.3	42.5	60.0	21.2	36.1
OWG (Ours)	GPT-4v	83.3	80.1	45.7	55.4	78.8	90.3	59.4	70.4

Table 2: Segmentation - mIoU(%) results in different language input types for cluttered indoor scenes from OCID.

target object and the rest segmented objects’ mask centers. To close the loop with vision, we further augment the library with two visual primitives, for open-vocabulary object detection and referring grounding. We utilise the ViLD open-vocabulary object detector, which is being prompted with a list of all object categories included in our experiments. In order to ground referring expressions, we allow GPT-4v to invoke CLIP in order to rank the output of the object detector according to the query, as in the CLIP-based baselines of our grounding experiments. Finally, in order to execute the actions, we use GR-ConvNet for grasp synthesis and a motion planner for moving the arm, as in the OWG pipeline. In all methods, three total attempts in grasping are allowed before counting the trial as failed.

## C Instance segmentation examples

In Fig 6, we visualize segmentation masks extracted from different methods for cluttered scenes from the OCID dataset, used in our open-ended grounding evaluation. We compare ground-truth masks with SAM, ViLD-RPN and UOIS methods, as explained in the paper. We observe that SAM tends to over-segment, however, we did not experiment with different hyper-parameter settings and use the automatic mask generator out-of-the-box. Suitable fine-tuning of SAM might provide crisp segmentations, as shown in other works. Both the RPN of the ViLD and the UOIS methods provide reasonable segmentation masks, with most dominant failures being like under-segmenting in cases of heavy clutter.

## D Open-ended grounding experiments

### D.1 Dataset Details

We manually annotate 173 images from OCID dataset with the following query types: a) **name** (open-vocabulary object descriptions), b) **attribute**, c) **spatial relations**, d) **visual relations**, e) **semantic relations**, f), **multi-hop reasoning**, and g) **user-affordances**. The number of annotations per query type given in Table 1. We make sure to include unique test scenes from the dataset and include images with heavy clutter. The target of each scene within a query type is unique, and we make sure to include images with distractor objects (of the same category as the target) for all query types that require relational reasoning (all except name and affordance).

The need for manual annotations to exhaust all possible language query inputs, as well as the need for manual testing via online demo applications for the considered specialist end-to-end methods (SEEM, PolyFormer) restrained us from conducting experiments in large-scale. Instead, we originally conducted experiments in a smaller subset of 52 images. Results are given in Table 2. Results follow similar patterns to the larger test set of the main paper. Specialist models (SEEM, PolyFormer) struggle with even simple name queries, scoring below 15% on average. This is potentially due to the high discrepancy between the training distribution of RefCOCO and Visual Genome and our test data, as well as the lack of relational and affordance-based language in these datasets. GPT-4v-based methods still compare favourably to CLIP-based baselines, even in the SoM setting where



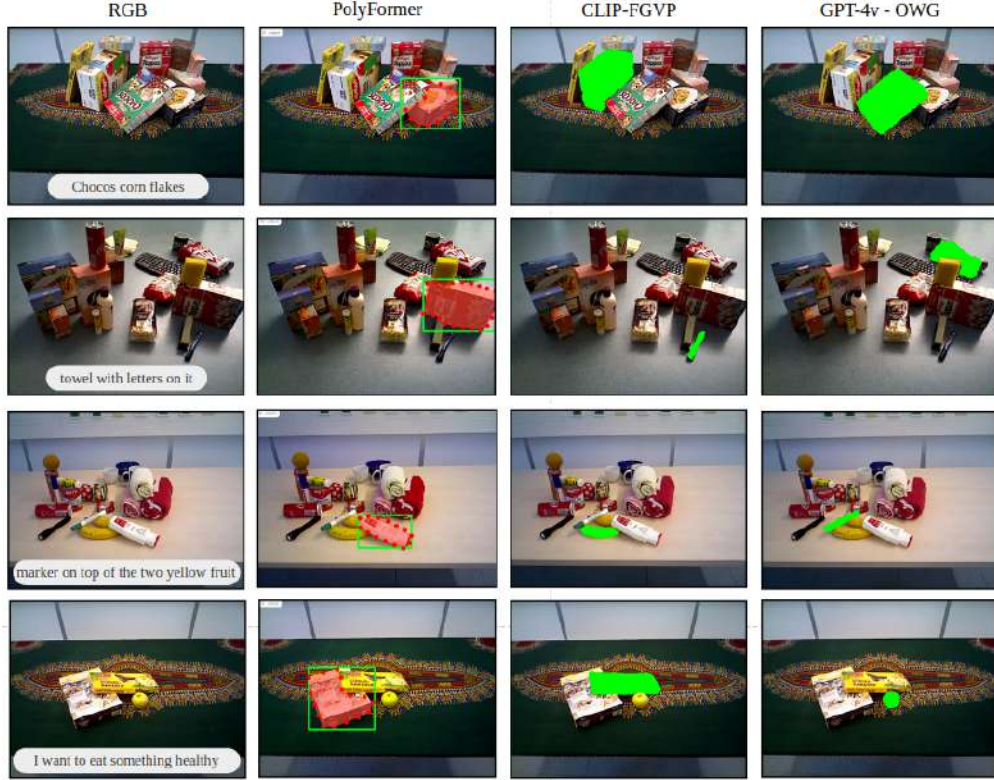


Figure 5: Example output segmentation masks of different grounding methods for OCID scenes.

single marked image is used. Overall, our OWG-grounder achieves an averaged mIoU score of 70.4%, which is almost  $\times 2$  from the previous approach. Regarding our custom FGVP-CLIP baseline (FGVP\*), we present analytical comparisons and ablation in the following subsection.

## D.2 Baselines Implementation and Ablations

We utilize the provided demo applications for the end-to-end methods (SEEM, PolyFormer) to conduct grounding experiments manually. For CLIP-based baselines, we re-implement all methods from the corresponding papers (ReCLIP, RedCircle, FGVP). We use the ViT-B visual encoder to extract features from image segments and the default BERT text encoder to represent the input query. CLIP-based baselines compute the cosine similarity between segment and text features to rank them and select the most similar segment as the final result via the argmax operator. Ground-truth masks

w/ Crop	w/ White-Back.	w/ Blur-Rev	w/ Gray-Rev	w/ Multi Temp.	Rect.	Ellipse	Mask	mIoU
							X	18.3
						X		31.1
					X			34.8
				X	X			33.7
			X		X			24.6
		X			X			26.3
		X	X		X			34.9
		X	X		X		X	41.5
		X	X	X	X		X	43.0
	X	X	X	X	X		X	<b>51.8</b>
X	X	X	X	X	X		X	51.2

Table 3: Component ablation studies for CLIP-based visual prompting. Results in %.

are used for all CLIP-based baselines, similar to GPT-4v ones. We would like to highlight that in the original papers, the aforementioned baselines use potential post-processing steps to enhance the grounding capabilities of CLIP. In particular, ReCLIP uses syntactic parsing to extract entity and relation words/phrases from the input query, as well as spatial relation resolution heuristics (e.g. 'left', 'on' etc. - designed specifically for the RefCOCO dataset) to process the relations analytically and combine CLIP predictions only for the entities. RedCircle and FGVP additionally utilize a "*subtraction*" post-processing step, where they further subtract from the similarity values the average in a set of mined hard-negative queries (again selected for a specific dataset). We believe that such steps constitute domain-aware hand-crafted efforts, which even though helpful, do not represent the challenges of open-ended generalization, which is the primary focus of this work. As a result, we do not consider such post-processing steps in our baseline implementation.

To further analyze the performance of CLIP-based baselines, we conduct ablation studies where we use specific elements of each method. In particular, we study: a) effect of using **multi-templates** for the text prompt, where we average text embeddings from multiple versions of the query, using templates from the original paper, b) averaging similarity scores from the visual prompt and **crops** of each segment, as originally proposed in ReCLIP, c) different visual prompt schemes, like drawing a boundary (**rectangle** or **ellipse** - as in RedCircle), converting to **grayscale** or **blurring** the rest of the frame (as proposed in FGVP), as well as a prompt that we discover ourselves works good, using a **white background** for the rest of the frame. We note that in our paper's results the element combinations we used are the following:

**ReCLIP**: rectangle prompt, multi-templates, blur-reverse + crop,

**RedCircle**: ellipse prompt, multi-templates, gray-reverse + blur-reverse,

**FGVP**: mask prompt, multi-templates, gray-reverse + blur-reverse

Ablation results are shown in Table 3. Our findings are the following: 1) drawing a rectangle prompt outperforms ellipse and mask (object contours) in itself, but ensembling rectangles and masks gives the best result, 2) using multiple text templates outperforms single-template only when ensembling multiple visual inputs, c) the most effective component is our method of replacing the rest of the frame with white background, compared to grayscale and reverse operators of FGVP, while ensembling all together gives the best performance. We call our custom FGVP baseline FGVP\*.

## References

- [1] Gpt-4v(ision) system card. 2023. URL <https://api.semanticscholar.org/CorpusID:263218031>.
- [2] J. Yang, H. Zhang, F. Li, X. Zou, C. yue Li, and J. Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *ArXiv*, abs/2310.11441, 2023. URL <https://api.semanticscholar.org/CorpusID:266149987>.
- [3] N. M. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538, 2017. URL <https://api.semanticscholar.org/CorpusID:12462234>.
- [4] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. hsin Chi, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022. URL <https://api.semanticscholar.org/CorpusID:247595263>.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, and et. al. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL <https://api.semanticscholar.org/CorpusID:218971783>.
- [6] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang. The dawn of lmms: Preliminary explorations with gpt-4v(ision). *ArXiv*, abs/2309.17421, 2023. URL <https://api.semanticscholar.org/CorpusID:263310951>.

## OCID Segmentation Mask comparisons



Figure 6: Comparison of different segmentation models for seven example cluttered indoor scenes from OCID dataset

[7] OpenAI. (mar 14 version) [large language model]. 2023.



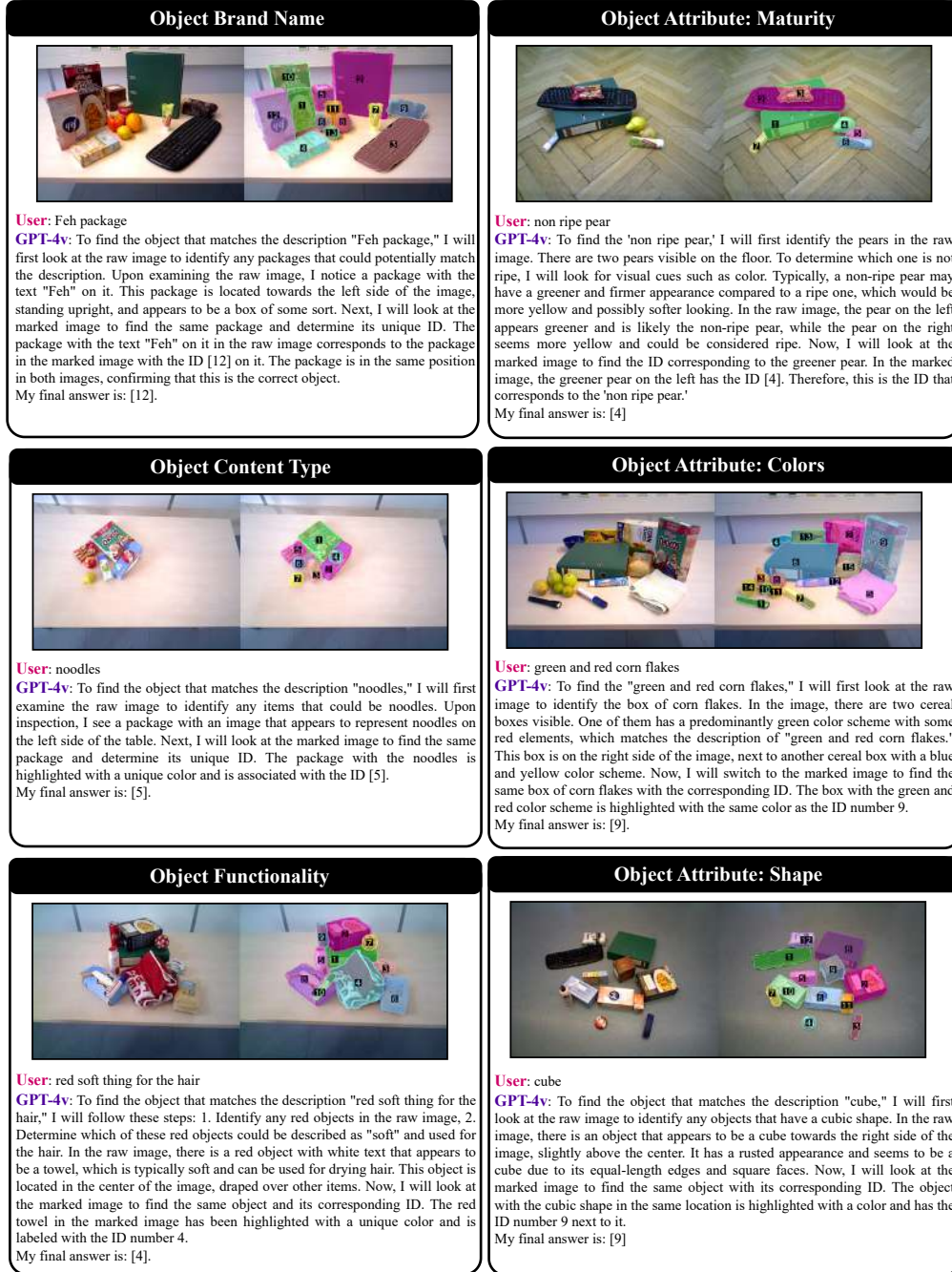


Figure 7: Example GPT-4v responses (1/3): object-related queries



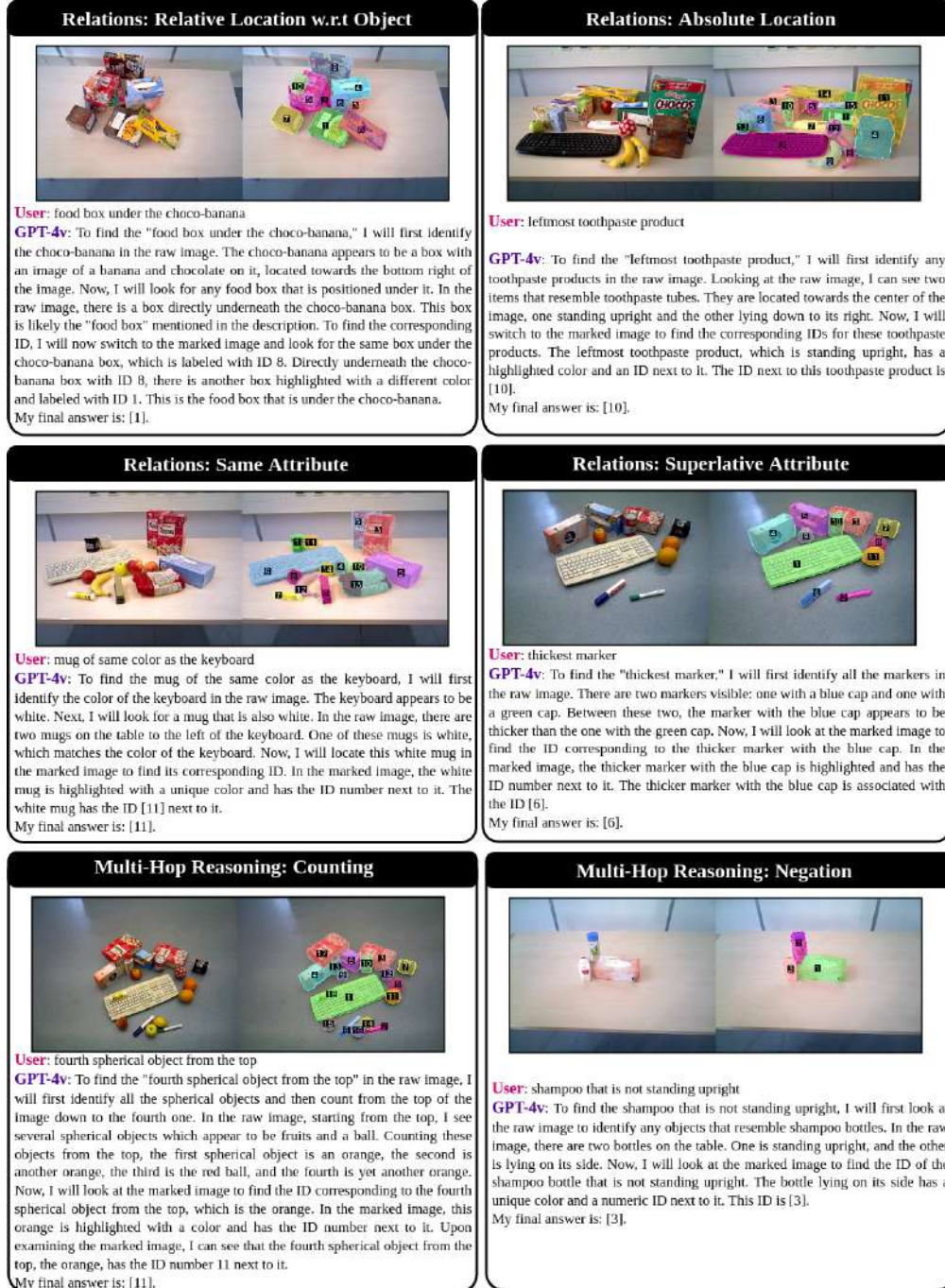


Figure 8: Example GPT-4v responses (2/3): relational reasoning queries

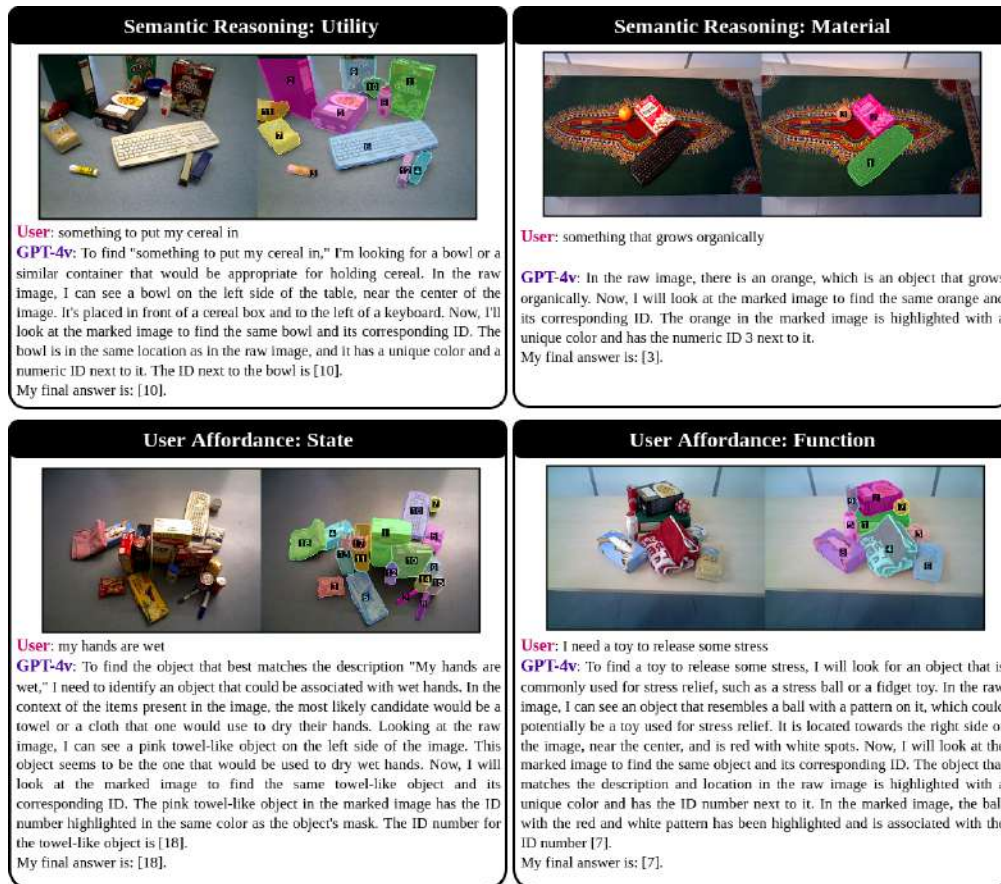


Figure 9: Example GPT-4v responses (3/3): semantics-related queries

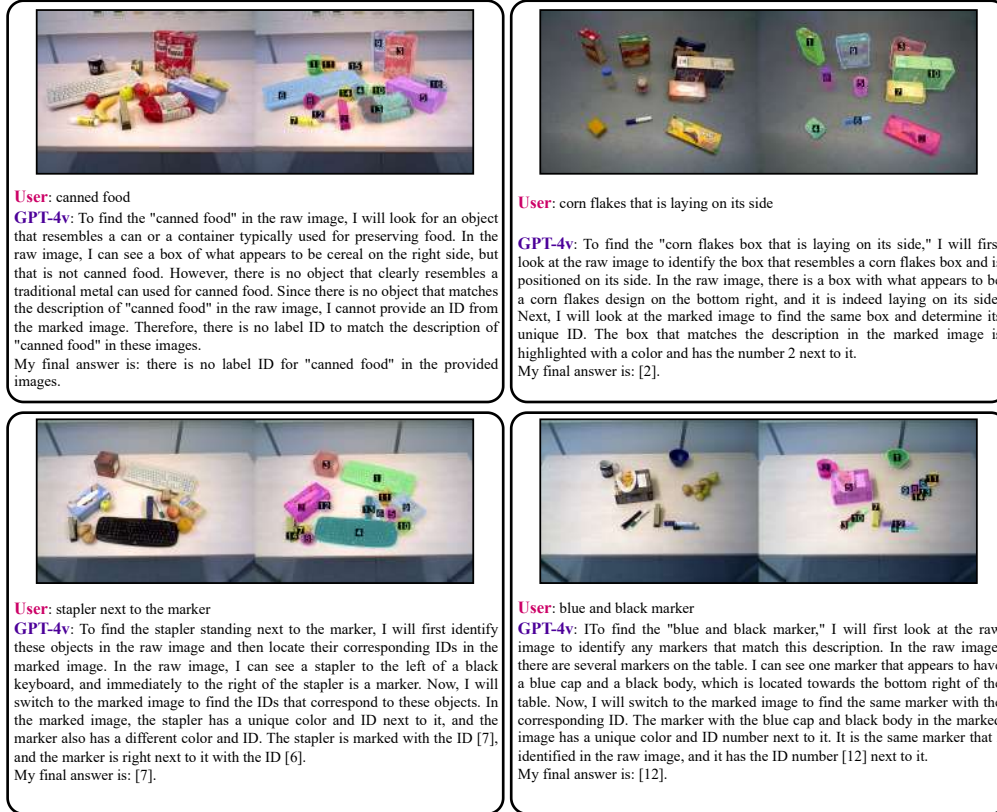


Figure 10: Example of GPT-4v response failures.