

Obstruction reasoning for robotic grasping

Supplementary Material

In this supplementary material, we provide more details regarding our benchmark UNOBench (Sec. A) and the training details of our method UNOGrasp (Sec. B). We also present more additional experimental details and analyses on our benchmark (Sec. C) and the real-robot evaluation (Sec. D). Finally we show a video (unograspRoboticResults.mp4) demonstrating the real-robot test with Qwen2.5-VL [4], Gemini Robotics-ER 1.5 [1] and UNOGrasp under Easy, Medium and Hard scenarios.

A. Additional details on UNOBench

A.1. Dataset construction and statistics

We construct the Synthetic subset of UNOBench starting from 8,007 scenes in MetaGraspNetV2 (37 viewpoints each) and apply a series of pre-processing steps. We first filter out scenes that are inappropriate for our benchmarking purpose: i) we remove empty scenes or scenes containing only a single object, *i.e.* no obstruction can be formed; ii) we remove scenes with obstructions that are not physically plausible (*e.g.*, objects penetrating each other due to simulation artifacts); iii) we discard scenes that contain bidirectional or cyclic obstruction patterns, as they will inevitably cause problematic obstruction paths.

After the filtering, we retain 6,255 scenes, from which we randomly sample four diverse viewpoints per scene, resulting in a final set of 25,020 images. Almost all objects serve as target objects for constructing the obstruction graphs and generating the VQA dataset. We exclude some objects when i) their obstruction ratios are below 1% as such light obstruction are often visually ambiguous and do not impede grasping execution, and ii) when their obstruction ratios above 95% as they become visually difficult to recognize due to such excessive obstruction. For the Real subset, we follow the same construction pipeline as in the Synthetic subset. As mentioned in Sec. 5, we split the synthetic scenes into training, validation, and testing sets with a 7:1:2 ratio, while all the real scenes are exclusively used for testing. Tab. A summarizes the dataset statistics of all subsets in UNOBench. Note that for the synthetic subset, some objects are annotated by human as “indescribable objects” as they are barely visible due to severe obstruction. We thus remove all VQA samples containing such objects in the NLP setting, resulting in fewer objects compared to the Oracle (SoM) setting. In addition, we randomly sample 2,000 out of 12,719 objects under Oracle (SoM) Test that have No Obstruction (No-Obs) for evaluation efficiency.

Table A. Statistics of the Oracle (SoM) and Natural Language Prompting (NLP) settings on both Synthetic and Real subsets.

Setting	Split	#Objects	No-Obs	Easy	Med	Hard
Synthetic						
Oracle (SoM)	Train	67,945	43,350	16,553	7,305	737
	Val	9,539	6,242	2,260	974	63
	Test	8,863	2,000	4,576	2,085	202
NLP	Train	61,690	40,389	14,801	5,969	531
	Val	8,785	5,853	2,056	830	46
	Test	8,526	1,993	4,477	1,936	180
Real (test only)						
Oracle (SoM)	Test	2,232	1,341	606	263	22
NLP	Test	2,232	1,341	606	263	22

A.2. Natural-language annotation pipeline

Fig. A presents a complete example from UNOBench, including the input image, the corresponding obstruction graph, pairwise obstruction relations, and the generated VQA samples in both the Oracle (SoM) and NLP settings. Below, we present the detailed procedure for obtaining the high-quality object names in natural language for the NLP setting.

LLM pre-annotation. We first perform an automated pre-annotation with a vision-language model (VLM), *i.e.*, gpt-4o. Specifically, given the SoM-labeled images, the VLM first produces one short natural-language description per object. We instruct the VLM to consider, especially in scenes containing multiple instances of the same semantic category, to produce names using spatial specifiers (*e.g.*, left, right, top, bottom) to uniquely distinguish multiple instances of the same category. Moreover, we instruct the output to follow a shared JSON format to ease the parsing at later stages.

Prolific human annotation. For scenes containing more than five objects, we further conduct human annotation on all objects, where the VLM-generated descriptions are revised and corrected by human annotators. To collect human annotations, we used a commercial platform, Prolific, for its high-quality annotation and ethical compliance. Specifically, we recruited annotators who are native or primary English speakers located in six English-speaking countries, with a $\geq 99\%$ historical approval rate on the platform, under the recommended pay rate. A total of 196 annotators participated. Each annotator could complete at most 45 scenes, and the study was designed to be finished within 80 minutes to prevent fatigue. The participant pool had a balanced gender distribution and was mainly from the UK (54%), the US (27%), and Canada (12%), with birth countries spanning 11 regions. Our annotations were collected via a web interface where we present to the annotators the scene with colored



Figure A. UNOBench example. Illustration of scene representation and corresponding VQA formulations.

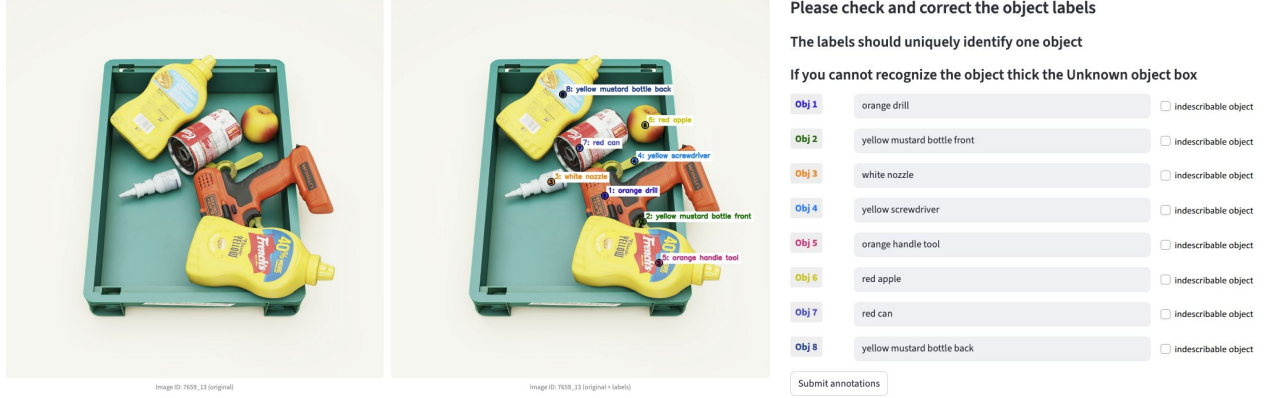


Figure B. Interface of the online annotation platform.

dots, each dot marks an object and its VLM-generated preliminary name in the same color next to the dot (as shown in Fig. B). Annotators are requested to verify whether each provided name uniquely and accurately referred to the marked object. In case no, they should correct the name accordingly. Under cases where objects are barely visible due to heavy obstruction, annotators can mark them as “indescribable object”.

Moreover, we further conducted a review on top of the collected annotations from Prolific, to further check for semantic or referential inconsistencies. In total, 5,400 images and 41,193 object names were examined; 4,78 images and 17,261 names were corrected, and the remaining names were confirmed to be accurate.

A.3. Evaluation metrics

As introduced in Sec. 3.1, we evaluate model predictions at three levels, including i) the outcome-level metrics in terms of how accurate we can identify top obstructors, ii) object-level reasoning in terms of how accurate we can predict pairwise obstruction relationships, and iii) path-level reasoning in terms of how accurate all obstruction paths are predicted. Algorithm 1 outlines the evaluation algorithm. For each sample, we extract the predicted top obstructors $\mathcal{F}_{\text{pred}}$ and predicted reasoning paths $\mathcal{A}_{\text{pred}}$ from `<answer>` and `<think>` respectively, and load the corresponding ground-truth sets \mathcal{F}_{gt} and \mathcal{A}_{gt} .

Outcome-level metrics. Since a target may have multiple top-level obstructors, we compute precision, recall, and F1 between $\mathcal{F}_{\text{pred}}$ and \mathcal{F}_{gt} to evaluate whether the model correctly identifies all required top-level blockers without missing or introducing wrong objects.

Object-level reasoning. We extract all pairwise obstruction triplets from predicted and ground truth paths to form T_{pred} and T_{gt} . OP/OR/F1_{rel} measure the accuracy of these fundamental obstruction edges, independent of path depth or multi-path structure. As shown in Tabs. 2 and 4, existing

VLMs still struggle with this basic reasoning ability.

Path-level reasoning. Let $\mathcal{P} = \{p_1, \dots, p_m\}$ and $\mathcal{G} = \{g_1, \dots, g_n\}$ denote the predicted and ground-truth path sets, respectively. For each pair (p_i, g_j) , we compute the normalized graph edit distance:

$$\text{NED}(p_i, g_j) = \frac{\text{Levenshtein}(p_i, g_j)}{\max(|p_i|, |g_j|)} \in [0, 1].$$

This distance captures three fundamental reasoning errors (Fig. 2): substitutions (incorrect relations), deletions (missing objects), and insertions (extra objects). If $m < n$ (under prediction), we append $(n - m)$ dummy predictions with a fixed missing penalty (weight set to 1); if $m > n$ (over-prediction), we append $(m - n)$ dummy ground-truth paths with a redundant penalty (also 1). We then apply the Hungarian algorithm to find the minimum-cost matching between \mathcal{P} and \mathcal{G} , and report the average matching cost as MP_NED, which measures full multi-path consistency.

B. Additional details on UNOGrasp

B.1. SFT details

To warm-start the model’s obstruction reasoning capability, we first optimize f_{Θ} with supervised fine-tuning (SFT) on the synthetic set of UNOBench. Each training example is represented as (I, q, r, a) , where I is the RGB observation, q is the free-form instruction referring to the target object o_t , r is the visually grounded reasoning chain describing the ancestor objects $\mathcal{A}(o_t)$, and a is the final prediction corresponding to the top-level obstructors $\mathcal{F}(o_t)$. We concatenate r and a into a single output sequence $y = (y_1, \dots, y_{T_{r,a}})$.

The objective of SFT is to maximize the likelihood of generating both r and a conditioned on (I, q) . Using the standard autoregressive formulation, the training loss is

Algorithm 1 Evaluation Algorithm

Require: Predictions \mathcal{P} , Ground-truth \mathcal{G} **Ensure:** SR-P/R/F1, OP/OR/F1, MP_NED**1: Parse and Load Data****for** each sample (p, g) in $(\mathcal{P}, \mathcal{G})$ **do** $\mathcal{F}_{\text{pred}}, \mathcal{A}_{\text{pred}} \leftarrow$ Extract sets from $p.\text{<answer>}$ and $p.\text{<think>}$ $\mathcal{F}_{\text{gt}}, \mathcal{A}_{\text{gt}} \leftarrow$ Load $g[\text{top_objects}]$ and $g[\text{obs_paths}]$ **2: Outcome-level Metrics** $(SR_P, SR_R, SR_F1) \leftarrow \text{ComputePRF}(\mathcal{F}_{\text{pred}}, \mathcal{F}_{\text{gt}})$ **3: Reasoning-level Metrics****Object-level Reasoning:** $T_{\text{pred}} \leftarrow$ Extract triplets (obj_a, rel, obj_b) from $\mathcal{A}_{\text{pred}}$ $T_{\text{gt}} \leftarrow$ Extract triplets from \mathcal{A}_{gt} $(OP, OR, F1_{\text{rel}}) \leftarrow \text{ComputePRF}(T_{\text{pred}}, T_{\text{gt}})$ **Path-level Reasoning (MP_NED):** $m, n \leftarrow |\mathcal{A}_{\text{pred}}|, |\mathcal{A}_{\text{gt}}|$ Initialize cost matrix $C \in \mathbb{R}^{m \times n}$ **for** $i \leftarrow 1$ to m , $j \leftarrow 1$ to n **do** $C_{ij} \leftarrow \text{Levenshtein}(p_i, g_j) / \max(|p_i|, |g_j|)$ **end for**matches $\leftarrow \text{HungarianMatch}(C)$ MP_NED $\leftarrow \frac{1}{\max(m, n)} \sum_{(i, j) \in \text{matches}} C_{ij}$

Store all computed metrics

end for**return** Mean metrics over all samples

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(I, q, r, a) \sim \mathcal{D}} \left[\sum_{t=1}^{T_{r, a}} \log p_{\Theta}(y_t \mid I, q, y_{<t}) \right] \quad (6)$$

where \mathcal{D} denotes the SFT training set, and p_{Θ} is the token-level conditional distribution produced by f_{Θ} . This supervision encourages the model to output step-by-step obstruction chains that are physically grounded (*i.e.*, each step corresponds to a contacting neighbor) and to predict the correct set of top-level obstructors. The best resulting model f_{Θ}^{SFT} with obstruction information serves as the initialization for the subsequent reinforcement fine-tuning (RFT) stage, ensuring stable optimization under task-specific rewards.

B.2. RFT details

Starting from the best SFT-trained model f_{Θ}^{SFT} , we further optimize it with reinforcement fine-tuning using Group Relative Policy Optimization (GRPO). For each training prompt (I, q) sampled from the RFT dataset \mathcal{D} , the current policy $\pi_{\Theta}(\cdot \mid I, q)$ (induced by f_{Θ}^{SFT}) generates G candidate outputs $\{y^{(g)}\}_{g=1}^G$, where each $y^{(g)} = (y_1^{(g)}, \dots, y_{T(g)}^{(g)})$ is a full `<think>+<answer>` sequence. Each generated sample receives a reward composed of format and task reward:

$$r^{(g)} = \lambda_{\text{fmt}} r_{\text{fmt}}^{(g)} + \lambda_{\text{task}} r_{\text{task}}^{(g)},$$

as defined in the main paper. GRPO uses group-wise relative advantages by subtracting the within-group mean reward,

$$\bar{r} = \frac{1}{G} \sum_{g=1}^G r^{(g)}, \quad A^{(g)} = r^{(g)} - \bar{r}.$$

The RFT objective then minimizes the following loss:

$$\mathcal{L}_{\text{RFT}}(\Theta) = -\mathbb{E}_{(I, q) \sim \mathcal{D}} \left[\frac{1}{G} \sum_{g=1}^G A^{(g)} \sum_{t=1}^{T^{(g)}} \log \pi_{\Theta}(y_t^{(g)} \mid I, q, y_{<t}^{(g)}) \right] + \beta \text{KL}(\pi_{\Theta} \parallel \pi_{\Theta}^{\text{SFT}}), \quad (7)$$

where π_{Θ_0} is the frozen SFT reference policy initialized from f_{Θ_0} , and β controls the strength of KL regularization. The first term encourages generations with higher relative rewards within each group, while the KL term keeps the updated policy close to the SFT initialization, stabilizing optimization under task-specific rewards.

C. Additional experimental details and analysis**C.1. Implementation details**

UNOGrasp. Our method training is built on the Qwen official repo [4] for SFT, and VLM-R1 [22] for GRPO. For clarity, we list the training hyperparameters used in SFT and RFT in Tab. B and Tab. C, respectively.

Table B. Supervised Fine-Tuning

Hyperparameter	Value
Model	Qwen2.5-VL-3B-Instruct
Training epochs	2
Learning rate	1e-5
Batch size (per device)	4
Gradient accumulation	1
Precision	bf16
LR scheduler	Cosine
Warmup ratio	0.03
Gradient clipping	1.0
Vision tuning	Frozen
MM-MLP / LLM tuning	Enabled
Max sequence length	8192
Max image pixels	12.8M
DeepSpeed config	ZeRO-3

In Context Learning (ICL). We evaluate ICL for both Qwen2.5-VL and Gemini Robotics-ER 1.5 under four settings: Oracle (SoM) on Real scenes, NLP on Real scenes, Oracle (SoM) on Synthetic scenes, and NLP on Synthetic scenes. For each setting, three examples are randomly sampled from the corresponding subset (Synthetic or Real) to construct the few-shot ICL context. In the Oracle (SoM) setting, both models receive exactly the same ICL examples (Fig. C, D). The only difference arises in the Natural

Table C. Reinforcement Fine-Tuning (GRPO)

Hyperparameter	Value
Training epochs	1
Batch size (per device)	8
Gradient accumulation	2
Precision	bf16
Group size (G)	4
Max completion length	512
β (KL penalty)	0.04
Temperature	1.0
Top- p	0.9
Attention backend	FlashAttention-2
Gradient checkpointing	Enabled
Random seed	42
DeepSpeed config	ZeRO-2

Language Prompting setting: Qwen2.5-VL requires absolute pixel coordinates (x, y) (Fig. E, F), whereas Gemini Robotics-ER 1.5, based on their cookbook, uses normalized coordinates (y, x) within the range $[0, 1000]$ (Fig. G, H).

C.2. Additional analysis

Target-centric vs. scene-level graphs.

We compare two graph formulations for obstruction reasoning: (i) scene-level graphs that model relations among all objects, and (ii) target-centric graphs that focus only on objects affecting the accessibility of the queried target.

We adopt the target-centric formulation based on empirical observations. Using a target-centric graph as an inductive bias encourages the model to reason about obstructions that affect the target’s accessibility, leading to more consistent and interpretable predictions.

In contrast, preliminary experiments with scene-level graphs often exhibit *obstruction hallucinations*, where predicted obstruction paths include non-adjacent or irrelevant objects. This suggests that modeling all pairwise relations introduces unnecessary complexity and distracts the reasoning process from the target.

These observations motivate our design choice of using a compact target-centric obstruction graph in UNOGrasp.

Comparison with specialized grasp planners

We compare UNOGrasp with specialized grasp planning (Tab. D), including ThinkGrasp [17] and FreeGrasp [12].

We report SR-Precision (SR-P) as these methods are designed to predict the primary obstructor in a single step. ThinkGrasp achieves better performance than FreeGrasp on Hard cases, likely due to its design for handling invisible targets by reasoning about topmost objects.

In contrast, UNOGrasp consistently outperforms both baselines across all difficulty levels, demonstrating stronger robustness in obstruction reasoning.

Table D. Comparison with specialized grasp planning methods on UNOBench.

Method	No obs.	Easy	Med.	Hard
	SR (%) \uparrow	SR-P \uparrow	SR-P \uparrow	SR-P \uparrow
ThinkGrasp	34.0	15.1	17.8	27.2
FreeGrasp	31.9	62.4	45.6	18.2
UNOGrasp	70.0	71.1	62.9	54.5

Complete ablations for all variants

Tab. E and Fig. I present the numerical and visual results of incorporating different types of obstruction information during SFT across the Easy, Medium, Hard, and Overall setting. All obstruction cues consistently improve the baseline model’s obstruction reasoning performance, and the improvement becomes more pronounced as the scene difficulty increases. Among them, obstruction ratio yields the strongest boost, achieving a 5.8% increase in SR-F1 under the Hard setting, indicating that continuous ratio signals help learning complex obstruction reasoning.

We further combine the two most effective types of obstruction information, *i.e.*, Obstruction Ratio and Contact Point, and evaluate both full-sentence and short-sentence templates (as shown in Fig. J). The short one achieves better performance generally, especially in Hard scenes. We hypothesize that this might be because shorter templates reduce unnecessary tokens, allowing the model to focus more on critical obstruction cues and thus produce higher quality reasoning in long and complex obstruction paths. However, in the SoM setting, the benefits of joining Obstruction Ratio with Contact Point are not better, suggesting the need for better fusion strategies.

More qualitative results.

We present qualitative results on the UNOBench real set, comparing UNOGrasp, Gemini Robotics-ER 1.5 (ICL), and Qwen2.5-VL (ICL). Notably, our method does not use any real-set data during training, whereas Gemini and Qwen are each provided with three real few-shot examples covering *no-obstruction*, *single-path*, and *multi-path* ICL examples.

Fig. K shows no-obstruction examples. Even though the ICL context explicitly includes a similar case, both Gemini Robotics-ER 1.5 and Qwen2.5-VL hallucinate the obstruction among objects that are far apart. In contrast, with grounded obstruction information, UNOGrasp effectively suppresses such hallucinations. Additionally, we observe that for object pairs involved in an obstruction relation, Gemini Robotics-ER 1.5 and Qwen2.5-VL tend to regard the queried target as the *object being obstructed*, even when the target is actually the one *doing the obstructing*.

Fig. L presents Easy examples. UNOGrasp and Gemini Robotics-ER 1.5 both produce mostly correct predictions in this case, but Gemini is still prone to misidentifying spatially close objects as obstructors. On the other hand, Qwen2.5-

System Prompt

You are an assistant for robotic grasp planning. When asked which object must be removed first to grasp a specific object:

- If the target object is not obstructed, return the target object's ID itself.
- If the object has one obstruction path, reason step-by-step along that path and end with the top-most object that must be removed first.
- If the object has multiple obstruction paths, reason step-by-step for each path separately.

In the <answer>...</answer> tag, output ALL distinct top-most objects as a JSON list.

Use <think>...</think> tags for your reasoning, and put ONLY the final object IDs in <answer>...</answer>.

Examples

Example 1



Image: Image 1

Question: Which object must be removed first to grasp object 1?

Answer:

<think>Object 1 is not obstructed.</think>

<answer>[1]</answer>

Example 2

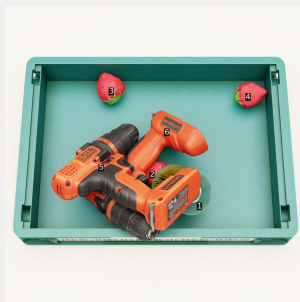


Image: Image 2

Question: Which object must be removed first to grasp object 1?

Answer:

<think>

Object 1 is obstructed by object 5; Object 5 is not obstructed.

</think>

<answer>[5]</answer>

Example 3

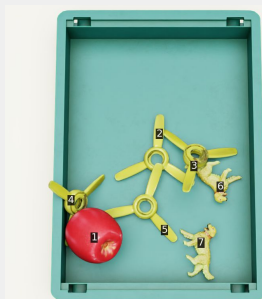


Image: Image 3

Question: I want to grasp object 2. What should I remove first?

Answer:

<think>

Path1: Object 2 is obstructed by object 3; Object 3 is not obstructed.

Path2: Object 2 is obstructed by object 5; Object 5 is obstructed by object 1; Object 1 is not obstructed.

</think>

<answer>[1, 3]</answer>

Figure C. Synthetic set Oracle (SoM) examples for Qwen2.5-VL (ICL) and Gemini Robotics-ER 1.5 (ICL).

System Prompt

You are an assistant for robotic grasp planning. When asked which object must be removed first to grasp a specific object:

- If the target object is not obstructed, return the target object's ID itself.
- If the object has one obstruction path, reason step-by-step along that path and end with the top-most object that must be removed first.
- If the object has multiple obstruction paths, reason step-by-step for each path separately.

In the <answer>...</answer> tag, output ALL distinct top-most objects as a JSON list.

Use <think>...</think> tags for your reasoning, and put ONLY the final object IDs in <answer>...</answer>.

Examples

Example 1



Image: Image 1

Question: Before I grasp object 3, what must be removed?

Answer:

<think>Object 3 is not obstructed.</think>

<answer>[3]</answer>

Example 2



Image: Image 2

Question: Which object must be removed first to grasp object 3?

Answer:

<think>

Object 3 is obstructed by object 6; Object 6 is not obstructed.

</think>

<answer>[6]</answer>

Example 3



Image: Image 3

Question: I want to grasp object 4. What should I remove first?

Answer:

<think>

Path1: Object 4 is obstructed by object 1; Object 1 is not obstructed.

Path2: Object 4 is obstructed by object 5; Object 5 is not obstructed.

</think>

<answer>[1, 5]</answer>

Figure D. Real set Oracle (SoM) examples for Qwen2.5-VL (ICL) and Gemini Robotics-ER 1.5 (ICL).

System Prompt

You are an assistant specialized in robotic grasp planning based on obstruction reasoning.
When asked which object must be removed first to grasp a specific target object in a single image:

- If the target object is not obstructed, return the target object's name/description and its coordinates.
- If the target object has one obstruction path, reason step by step along that path until you find the top-most occluding object that must be removed first.
- Each reasoning step must reference objects with explicit (x,y) coordinates.
- If the target object has multiple obstruction paths, reason step by step for each path separately, and include all distinct top-most occluding objects in the final answer.
- All reasoning must be enclosed within a single pair of <think>...</think> tags.
- The final answer must be enclosed in <answer>...</answer> tags and formatted strictly as:
<answer>[<points x y>object name</points>, ...]</answer>

Examples

Example 1



Image: Image 1

Question: What is the top object that occludes red snack can?

Answer:

<think>red snack can at (426, 355) is not obstructed.</think>

<answer>[<points 426 355>red snack can</points>]</answer>

Example 2



Image: Image 2

Question: Which object must be removed first to grasp air filter?

Answer:

<think>

air filter at (743, 760) is obstructed by left drill at (418, 632); left drill at (418, 632) is not obstructed.

</think>

<answer>[<points 418 632>left drill</points>]</answer>

Example 3



Image: Image 3

Question: I want to grasp top yellow propeller. What should I remove first?

Answer:

<think>

Path1: top yellow propeller at (600, 532) is obstructed by right yellow propeller at (687, 613); right yellow propeller at (687, 613) is not obstructed.

Path2: top yellow propeller at (600, 532) is obstructed by bottom yellow propeller at (613, 774); bottom yellow propeller at (613, 774) is obstructed by red apple at (436, 793); red apple at (436, 793) is not obstructed.

</think>

<answer>[<points 436 793>red apple</points>, <points 687 613>right yellow propeller</points>]</answer>

Figure E. Synthetic set Natural Language Prompting examples for Qwen2.5-VL (ICL).

System Prompt

You are an assistant specialized in robotic grasp planning based on obstruction reasoning.
When asked which object must be removed first to grasp a specific target object in a single image:

- If the target object is not obstructed, return the target object's name/description and its coordinates.
- If the target object has one obstruction path, reason step by step along that path until you find the top-most occluding object that must be removed first.
- Each reasoning step must reference objects with explicit (x,y) coordinates.
- If the target object has multiple obstruction paths, reason step by step for each path separately, and include all distinct top-most occluding objects in the final answer.
- All reasoning must be enclosed within a single pair of <think>...</think> tags.
- The final answer must be enclosed in <answer>...</answer> tags and formatted strictly as:
<answer>[<points x y>object name</points>, ...]</answer>

Examples

Example 1



Image: Image 1

Question: Before I grasp blue container on the left, what must be removed?

Answer:

<think>

blue container on the left at (662, 746) is not obstructed.

</think>

<answer>[<points 662 746>blue container on the left</points>]</answer>

Example 2



Image: Image 2

Question: What is the top object that occludes middle right bottle?

Answer:

<think>

middle right bottle at (1129, 578) is obstructed by top right box at (1322, 540); top right box at (1322, 540) is not obstructed.

</think>

<answer>[<points 1322 540>top right box</points>]</answer>

Example 3



Image: Image 3

Question: I want to grasp the top right cup. What should I remove first?

Answer:

<think>

Path1: top right cup at (1118, 410) is obstructed by center cup at (1032, 574); center cup at (1032, 574) is not obstructed.

Path2: top right cup at (1118, 410) is obstructed by top knife at (699, 441); top knife at (699, 441) is not obstructed.

</think>

<answer>[<points 1032 574>top cup</points>, <points 699 441>top knife</points>]</answer>

Figure F. Real set Natural Language Prompting examples for Qwen2.5-VL (ICL).

System Prompt

You are an assistant specialized in robotic grasp planning based on obstruction reasoning.
When asked which object must be removed first to grasp a specific target object in a single image:

- If the target object is not obstructed, return the target object's name/description and its coordinates.
- If the target object has one obstruction path, reason step by step along that path until you find the top-most occluding object that must be removed first.
Each reasoning step must reference objects with explicit (x,y) coordinates.
- If the target object has multiple obstruction paths, reason step by step for each path separately, and include all distinct top-most occluding objects in the final answer.
- All reasoning must be enclosed within a single pair of <think>...</think> tags.
- The final answer must be enclosed in <answer>...</answer> tags and formatted strictly as:
[{"point": [y, x], "label": "object name"}], where the coordinates are normalized between 0–1000.

Examples

Example 1



Image: Image 1

Question: What is the top object that occludes red snack can?

Answer:

<think>

red snack can at (296, 355) is not obstructed.

</think>

<answer>[{"point": [296, 355], "label": "red snack can"}]</answer>

Example 2



Image: Image 2

Question: Which object must be removed first to grasp air filter?

Answer:

<think>

air filter at (633, 619) is obstructed by left drill at (527, 348); left drill at (527, 348) is not obstructed.

</think>

<answer>[{"point": [527, 348], "label": "left drill"}]</answer>

Example 3



Image: Image 3

Question: I want to grasp top yellow propeller. What should I remove first?

Answer:

<think>

Path1: top yellow propeller at (443, 500) is obstructed by right yellow propeller at (511, 573); right yellow propeller at (511, 573) is not obstructed.

Path2: top yellow propeller at (443, 500) is obstructed by bottom yellow propeller at (645, 511); bottom yellow propeller at (645, 511) is obstructed by red apple at (661, 363); red apple at (661, 363) is not obstructed.

</think>

<answer>[{"point": [661, 363], "label": "red apple"}, {"point": [511, 573], "label": "right yellow propeller"}]</answer>

Figure G. Synthetic set Natural Language Prompting examples for Gemini Robotics-ER 1.5 (ICL).

System Prompt

You are an assistant specialized in robotic grasp planning based on obstruction reasoning.
When asked which object must be removed first to grasp a specific target object in a single image:

- If the target object is not obstructed, return the target object's name/description and its coordinates.
 - If the target object has one obstruction path, reason step by step along that path until you find the top-most occluding object that must be removed first.
- Each reasoning step must reference objects with explicit (x,y) coordinates.
- If the target object has multiple obstruction paths, reason step by step for each path separately, and include all distinct top-most occluding objects in the final answer.
 - All reasoning must be enclosed within a single pair of `<think>...</think>` tags.
 - The final answer must be enclosed in `<answer>...</answer>` tags and formatted strictly as:
`[{"point": [y, x], "label": "object name"}]`, where the coordinates are normalized between 0–1000.

Examples

Example 1



Image: Image 1

Question: Before I grasp blue container on the left, what must be removed?

Answer:

`<think>`

blue container on the left at (621, 340) is not obstructed.

`</think>`

`<answer>[{"point": [621, 340], "label": "blue container on the left"}]</answer>`

Example 2



Image: Image 2

Question: What is the top object that occludes middle right bottle?

Answer:

`<think>`

middle right bottle at (481, 580) is obstructed by top right box at (450, 680);

top right box at (450, 680) is not obstructed.

`</think>`

`<answer>[{"point": [450, 680], "label": "top right box"}]</answer>`

Example 3



Image: Image 3

Question: I want to grasp top right cup. What should I remove first?

Answer:

`<think>`

Path1: top right cup at (342, 575) is obstructed by center cup at (478, 531); center cup at (478, 531) is not obstructed.

Path2: top right cup at (342, 575) is obstructed by top knife at (368, 360); top knife at (368, 360) is not obstructed.

`</think>`

`<answer>`

`[{"point": [478, 531], "label": "top cup"},`

`{"point": [368, 360], "label": "top knife"}]`

`</answer>`

Figure H. Real set Natural Language Prompting examples for Gemini Robotics-ER 1.5 (ICL).

Table E. Ablation on obstruction information in SFT (SOM variant). All SR-F1 and OR-F1 values are absolute (0–100 scale).

Method	Easy			Medium			Hard			Overall		
	SR-F1 ↑	OR-F1 ↑	MP-NED ↓	SR-F1 ↑	OR-F1 ↑	MP-NED ↓	SR-F1 ↑	OR-F1 ↑	MP-NED ↓	SR-F1 ↑	OR-F1 ↑	MP-NED ↓
Baseline (None)	80.1	79.2	0.125	65.0	58.5	0.396	44.3	45.7	0.543	74.7	71.9	0.220
+ Contact Point	81.0	80.1	0.120	65.2	58.5	0.394	48.7	45.0	0.540	75.3	72.5	0.216
+ degree word	80.9	79.8	0.125	65.1	59.2	0.388	44.7	44.8	0.541	75.1	72.5	0.217
+ Ratio	81.8	80.9	0.115	67.1	59.4	0.389	50.1	46.9	0.525	76.4	73.3	0.210
+ Ratio + Contact Point	81.2	80.3	0.118	66.5	58.7	0.389	47.1	47.0	0.529	75.7	72.7	0.212
+ Ratio + Contact Point (Short)	81.1	80.3	0.116	66.7	59.5	0.392	49.9	49.2	0.517	75.8	73.0	0.212

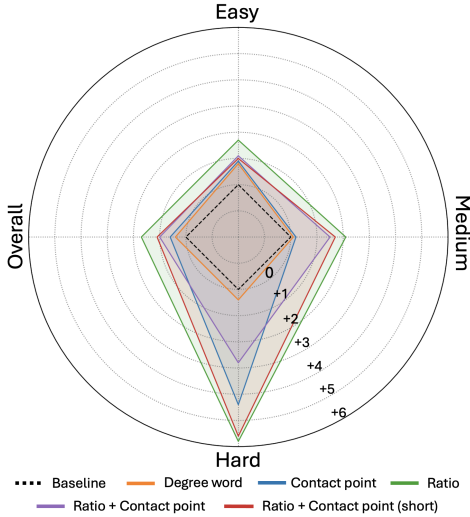


Figure I. Radar chart illustrating the ablation study of obstruction information used during SFT. We compare three difficulty levels (Easy, Medium, Hard) and overall with different obstruction cues.

VL exhibits more severe hallucinations: the predicted object names and coordinates are frequently misaligned, and the reasoning in `<think>` sometimes directly contradicts the final `<answer>` provided.

Fig. M illustrates more complex Medium cases with multiple obstruction paths or deeper reasoning chains. Here, UNOGrasp typically produces complete and coherent reasoning, tracing each path up to the correct top-level obstructor. Gemini Robotics-ER 1.5 tends to terminate the reasoning prematurely before reaching the true top obstructor, resulting in incorrect predictions.

Failure-case visualizations.

Fig. N summarizes several typical failure cases of UNOGrasp. The first category arises when objects are in physical contact but do not form an actual obstruction. In these cases, the model incorrectly predicts an obstruction relationship. Even when the model outputs a low occlusion ratio (e.g., 1%), indicating minimal obstruction, it still struggles to distinguish mere contact from true obstruction.

The second category occurs in scenes containing multiple objects with similar features (e.g., shapes and colors). When such objects touch and form obstructions, the model tend to fail at both the detection stage and the subsequent obstruction

reasoning stage, ultimately leading to incorrect identification of the obstruction paths.

D. Real-Robot Experiments

We conducted extensive real-world experiments comparing our approach against state-of-the-art VLM baselines. The experiments were designed to test obstruction reasoning in cluttered environments, specifically evaluating the generalization capability of VLMs in unseen real-world setups. We show the models’ outputs and the subsequent robotic executions under the Easy, Medium and Hard scenarios in the demonstration video. Here below, we describe in details the robotic setup and the experimental procedure.

D.1. Robotic Setup

The experimental platform consists of a Universal Robots UR5e 6-DoF manipulator equipped with a Robotiq 2F-85 parallel-jaw gripper. The observation is provided by a Stereolabs ZED 2 stereo camera mounted in a top-down configuration, approximately 0.8 m above the workspace. The workspace contains a bin populated with a diverse set of rigid objects (see Fig. O).

To ensure a fair comparison, the experimental framework integrates every VLM reasoning agent with an identical grasp generation backend. We utilize GraspNet [11] to generate a dense set of candidate 6-DoF grasp poses ($SE(3)$) globally across the scene, and GroundedSAM [19] to filter and select the specific poses associated with the object mask predicted by the VLM. This isolation ensures that performance differences are attributable solely to the reasoning capabilities of the tested VLMs, i.e. UNOGrasp, Gemini Robotics-ER 1.5, and Qwen2.5-VL.

D.2. Experimental Procedure

The experiments follow a step-wise protocol to ensure that all models are evaluated on identical scene configurations at each individual time step.

Initialization. For each scenario, the operator arranges the objects in the bin and initializes the system with three parameters: (1) the *User Prompt* (e.g., “grasp the red block”), which remains constant across all time steps; (2) the *Number of Objects* in the scene; and (3) the *Maximum Path Length* (k_{max}), defining the expected number of removal steps.

UnoGrasp System Prompt

```
BASE_SYSTEM_PROMPT = (  
    "You are an assistant for robotic grasp planning. When asked which object must be removed first "  
    "to grasp a specific object:\n\n"  
    "- If the target object is not obstructed, return the target object's ID itself.\n"  
    "- If the object has one obstruction path, reason step-by-step along that path and end with the "  
    "top-most object that must be removed first.\n"  
    "- If the object has multiple obstruction paths, reason step-by-step for each path separately. "  
    "In the <answer>...</answer> tag, output ALL distinct top-most objects as a JSON list.\n\n"  
    "Use <think>...</think> tags for your reasoning, and put ONLY the final object IDs in <answer>...</answer>."  
)  
  
SYSTEM_PROMPTS = {  
    "ratio_only": BASE_SYSTEM_PROMPT.replace(  
        "reason step-by-step along that path and end with the ",  
        "reason step-by-step along that path, include the occlusion ratio for each obstruction relation when  
available, and end with the "  
    ),  
    "degree_only": BASE_SYSTEM_PROMPT.replace(  
        "reason step-by-step along that path and end with the ",  
        "reason step-by-step along that path, describe each obstruction using natural-language severity terms such  
as slightly, partially, mostly, or heavily, and end with the "  
    ),  
    "point_only": BASE_SYSTEM_PROMPT.replace(  
        "reason step-by-step along that path and end with the ",  
        "reason step-by-step along that path, include the contact point using (x,y) coordinates for each obstruction  
relation when available, and end with the "  
    ),  
    "ratio_point": BASE_SYSTEM_PROMPT.replace(  
        "reason step-by-step along that path and end with the ",  
        "reason step-by-step along that path, include the occlusion ratio and contact point using (x,y) coordinates for  
each obstruction relation when available, and end with the "  
    ),  
    "ratio_point_short": BASE_SYSTEM_PROMPT.replace(  
        "reason step-by-step along that path and end with the ",  
        "reason step-by-step along that path, include the occlusion ratio and contact point using (x,y) coordinates for  
each obstruction relation when available, and end with the "  
    ),  
}
```

Reasoning step template

Base: Object {blocked} is obstructed by object {blocker}.

Ratio_only:

Object {blocked} is obstructed by object {blocker} with the occlusion ratio of {ratio*100:.0f}%.

Degree_only:

Object {blocked} is {sev} obstructed by object {blocker}.

Point_only:

Object {blocked} is obstructed by object {blocker} at the contact point ({px}, {py}).

Ratio_point:

Object {blocked} is obstructed by object {blocker} at the contact point ({px}, {py}) with the
occlusion ratio of {ratio*100:.0f}%.

Ratio_point_short:

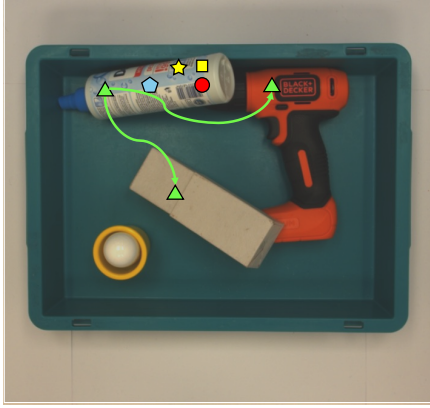
Object {blocked} is obstructed by object {blocker} with {ratio*100:.0f}% occlusion at ({px}, {py}).

Figure J. UNOGrasp system prompt templates and corresponding reasoning formats for obstruction information.

Execution Loop. The procedure operates in discrete time steps. At each step t , all VLMs will perform obstruction reasoning and predict the next move given same the physical state of the bin. The cycle is as follows:

1. **Observation:** The ZED 2 camera captures RGB-D data, which is normalized and cropped to the bin area.
2. **Obstruction Reasoning:** The VLM processes the visual data and the prompt to reason on the obstruction and predict the next object to be removed.
3. **Operator Feedback (Validation):** The human operator verifies if the predicted object by the VLM is a valid object that is reachable at the current step. If the prediction is not a top-level obstructor or the grasping point is on the bin itself, the robotic execution will be aborted. Crucially, if a VLM fails this validation, it will be excluded from the subsequent steps and marked as a failure.
4. **Execution:** If the prediction is validated, the robot executes the grasp using the integrated planner.
5. **Operator Tidy Up (Reset):** After the robot complete its execution, the operator manually restores the scene. Note that it is likely during the grasping operations, other objects are displaced accidentally. To ensure fair evaluation of VLMs on receiving the same scene observations at each time step, the operator resets the objects' arrangement to their original setup.

Once all VLMs have been evaluated for step t , the object identified as the correct obstruction is removed from the bin to advance the physical state to step $t + 1$. This loop repeats until the target object is grasped or all methods fail.



What is the first object I need to grasp to reach blue and white spray can?



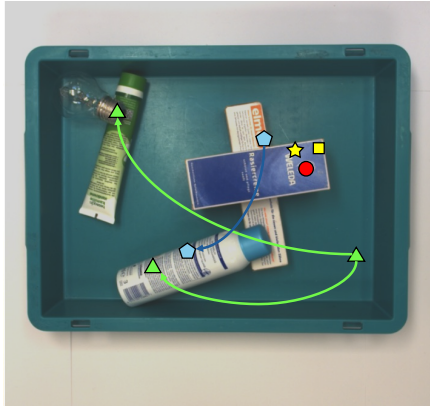
<think>
Path1: blue and white spray can at (248, 275) is obstructed by black+decker at (764, 284); black+decker at (764, 284) is not obstructed. Path2: blue and white spray can at (248, 275) is obstructed by cardboard at (458, 553); cardboard at (458, 553) is not obstructed.
</think>
<answer>[<points 248 275>blue and white spray can</points>, <points 458 553>cardboard</points>]**</answer>**



<think>
blue and white spray can at (218, 380) is not obstructed.
</think>
<answer>[{"point": [218, 380], "label": "blue and white spray can"}]**</answer>**



<think>
blue and white spray can at (560, 241) is not obstructed.
</think>
<answer>[<points 560 241>blue and white spray can</points>]**</answer>**



Which object must be removed first to grasp blue box?



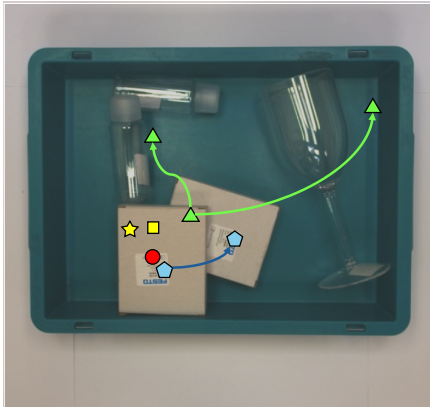
<think>
Path1: blue box at (1018, 746) is obstructed by tube at (309, 328); tube at (309, 328) is not obstructed. Path2: blue box at (1018, 746) is obstructed by tube at (421, 780); tube at (421, 780) is not obstructed.
</think>
<answer>[<points 1018 746>blue box</points>, <points 421 780>tube</points>]**</answer>**



<think>
blue box at (342, 575) is obstructed by white bottle at (621, 459); white bottle at (621, 459) is not obstructed.
</think>
<answer>[{"point": [621, 459], "label": "white bottle"}]**</answer>**



<think>
blue box at (842, 507) is not obstructed.
</think>
<answer>[<points 842 507>blue box</points>]**</answer>**



Before I grasp left box, what must be removed?



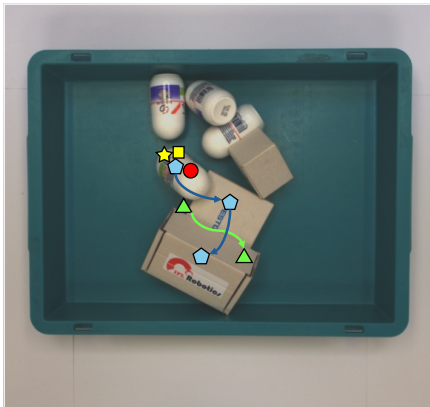
<think>
Path1: left box at (509, 630) is obstructed by bottle at (401, 388); bottle at (401, 388) is not obstructed. Path2: left box at (509, 630) is obstructed by glass at (1046, 297); glass at (1046, 297) is not obstructed.
</think>
<answer>[<points 401 388>bottle</points>, <points 1046 297>glass</points>]**</answer>**



<think>
left box at (636, 430) is obstructed by right box at (578, 510); right box at (578, 510) is not obstructed.
</think>
<answer>[{"point": [578, 510], "label": "right box"}]**</answer>**



<think>
left box at (430, 759) is not obstructed.
</think>
<answer>[<points 430 759>left box</points>]**</answer>**



What is the first object I need to grasp to reach bottom left bottle?



<think>
bottom left bottle at (662, 746) is obstructed by box at (500, 608); box at (500, 608) is not obstructed.
</think>
<answer>[<points 500 608>box</points>]**</answer>**

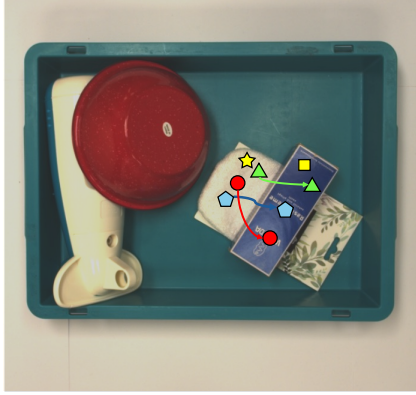


<think>
bottom left bottle at (417, 440) is obstructed by Festo box at (500, 510); Festo box at (500, 510) is obstructed by IFL Robotics box at (620, 470); IFL Robotics box at (620, 470) is not obstructed.
</think>
<answer>[{"point": [620, 470], "label": "IFL Robotics box"}]**</answer>**



<think>
bottom left bottle at (509, 508) is not obstructed.
</think>
<answer>[<points 509 508>bottom left bottle</points>]**</answer>**

Figure K. No-obstruction cases on the UNOBench real set.



To grasp white scrub pad, which object is on top of it?



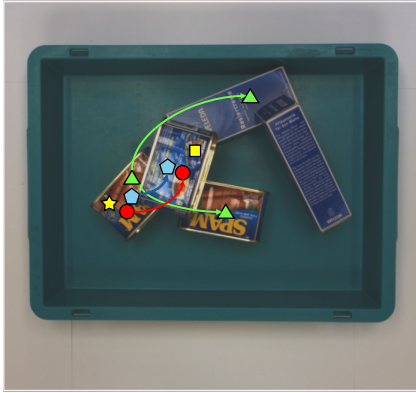
<think>
white scrub pad at (700, 550) is obstructed by box at (900, 560); box at (900, 560) is not obstructed.
</think>
<answer>[<points 900 560>box</points>]</answer>



<think>
white scrub pad at (511, 547) is obstructed by blue box at (500, 620); blue box at (500, 620) is not obstructed.
</think>
<answer>[{"point": [500, 620], "label": "blue box"}]</answer>



<think>
white scrub pad at (690, 587) is obstructed by blue rectangular box at (761, 728) with the occlusion ratio of 34%. blue rectangular box at (761, 728) is not obstructed.
</think>
<answer>[<points 761 728>blue rectangular box</points>]</answer>



Before I grasp bottom left can, what must be removed?



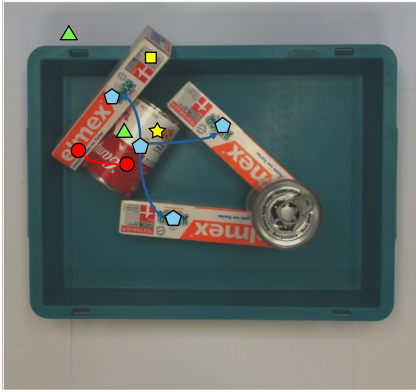
<think>
Path1: bottom left can at (348, 560) is obstructed by spam at (610, 660); spam at (610, 660) is not obstructed. Path2: bottom left can at (348, 560) is obstructed by weleda at (722, 309); weleda at (722, 309) is not obstructed.
</think>
<answer>[<points 610 660>spam</points>, <points 722 309>weleda</points>]</answer>



<think>
bottom left can at (498, 380) is obstructed by middle can at (417, 460); middle can at (417, 460) is not obstructed.
</think>
<answer>[{"point": [417, 460], "label": "middle can"}]</answer>



<think>
bottom left can at (329, 618) is obstructed by middle can at (504, 517) with the occlusion ratio of 29%. middle can at (504, 517) is not obstructed.
</think>
<answer>[<points 504 517>middle can</points>]</answer>



To pick up red and white can, which object is blocking it?



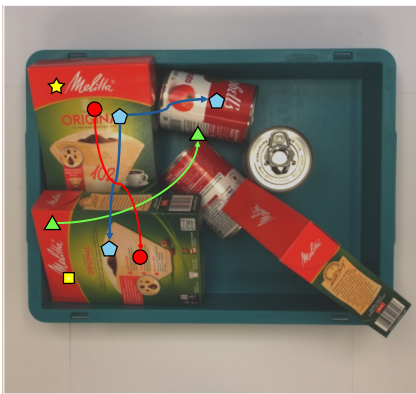
<think>
red and white can at (322, 408) is obstructed by top left box at (140, 100); top left box at (140, 100) is not obstructed.
</think>
<answer>[<points 140 100>top left box</points>]</answer>



<think>
Path1: red and white can at (370, 370) is obstructed by top left box at (260, 340); top left box at (260, 340) is not obstructed. Path2: red and white can at (370, 370) is obstructed by middle right box at (330, 520); middle right box at (330, 520) is not obstructed. Path3: red and white can at (370, 370) is obstructed by bottom right box at (560, 450); bottom right box at (560, 450) is not obstructed.
</think>
<answer>[{"point": [260, 340], "label": "top left box"}, {"point": [330, 520], "label": "middle right box"}, {"point": [560, 450], "label": "bottom right box"}]</answer>



<think>
red and white can at (340, 491) is obstructed by top toothpaste box at (208, 456) with the occlusion ratio of 17%. top toothpaste box at (208, 456) is not obstructed.
</think>
<answer>[<points 208 456>top toothpaste box</points>]</answer>



Which object must be removed first to grasp top large red box?



<think>
top large red box at (100, 684) is obstructed by can at (568, 410); can at (568, 410) is not obstructed.
</think>
<answer>[<points 568 410>can</points>]</answer>

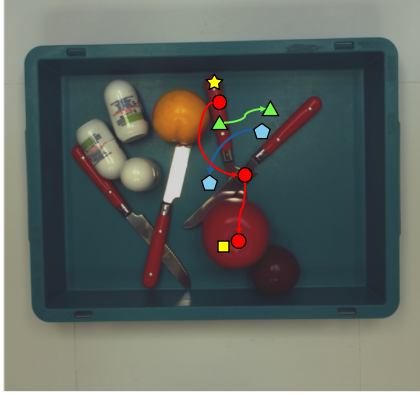


<think>
Path1: top large red box at (300, 350) is obstructed by top red can at (250, 500); top red can at (250, 500) is not obstructed. Path2: top large red box at (300, 350) is obstructed by bottom left red box at (620, 350); bottom left red box at (620, 350) is not obstructed.
</think>
<answer>[{"point": [250, 500], "label": "top red can"}, {"point": [620, 350], "label": "bottom left red box"}]</answer>



<think>
top large red box at (275, 312) is obstructed by bottom left coffee filter box at (400, 760) with the occlusion ratio of 8%. bottom left coffee filter box at (400, 760) is not obstructed.
</think>
<answer>[<points 400 760>bottom left coffee filter box</points>]</answer>

Figure L. Easy cases on the UNOBench real set.

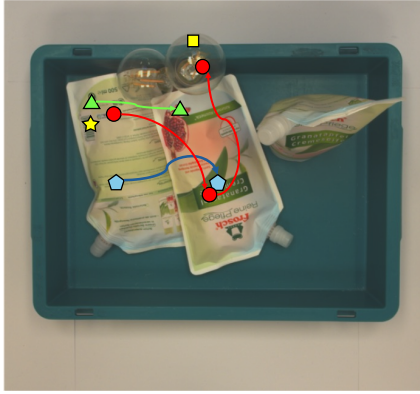


To grasp top right knife, which object is on top of it?

<think>
top right knife at (708, 350) is obstructed by top orange at (602, 396); top orange at (602, 396) is not obstructed.
</think>
<answer><points 602 396>top orange</points></answer>

<think>
top right knife at (330, 580) is obstructed by middle knife at (480, 500); middle knife at (480, 500) is not obstructed.
</think>
<answer>{{{"point": [480, 500], "label": "middle knife"}}</answer>

<think>
top right knife at (598, 273) is obstructed by bottom right knife at (694, 540) with the occlusion ratio of 1%. bottom right knife at (694, 540) is obstructed by bottom red tomato at (666, 723) with the occlusion ratio of 10%. bottom red tomato at (666, 723) is not obstructed.
</think>
<answer><points 666 723>bottom red tomato</points></answer>

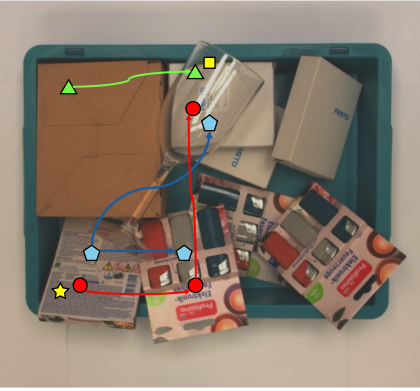


What is the top object that occludes left cream pouch?

<think>
left cream pouch at (230, 342) is obstructed by top cream pouch at (520, 342); top cream pouch at (520, 342) is not obstructed.
</think>
<answer><points 520 342>top cream pouch</points></answer>

<think>
left cream pouch at (490, 340) is obstructed by middle cream pouch at (500, 500); middle cream pouch at (500, 500) is not obstructed.
</think>
<answer>{{{"point": [500, 500], "label": "middle cream pouch"}}</answer>

<think>
left cream pouch at (291, 348) is obstructed by middle cream pouch at (607, 625) with the occlusion ratio of 2%. middle cream pouch at (607, 625) is obstructed by right light bulb at (539, 198) with the occlusion ratio of 4%. right light bulb at (539, 198) is not obstructed.
</think>
<answer><points 539 198>right light bulb</points></answer>

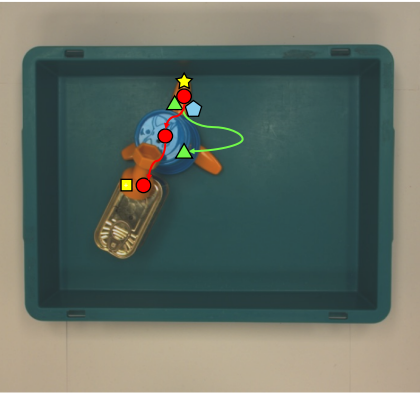


Tell me the top-most object that blocks access to bottom left box.

<think>
bottom left box at (162, 299) is obstructed by glass at (570, 239); glass at (570, 239) is not obstructed.
</think>
<answer><points 570 239>glass</points></answer>

<think>
Path1: bottom left box at (670, 320) is obstructed by wine glass at (300, 500); wine glass at (300, 500) is not obstructed. Path2: bottom left box at (670, 320) is obstructed by lighter package at (650, 480); lighter package at (650, 480) is not obstructed.
</think>
<answer>{{{"point": [300, 500], "label": "wine glass"}},{{{"point": [650, 480], "label": "lighter package"}}}</answer>

<think>
bottom left box at (219, 870) is obstructed by middle lighter pack at (567, 893) with the occlusion ratio of 1%. middle lighter pack at (567, 893) is obstructed by wine glass at (564, 339) with the occlusion ratio of 4%. wine glass at (564, 339) is not obstructed.
</think>
<answer><points 564 339>wine glass</points></answer>



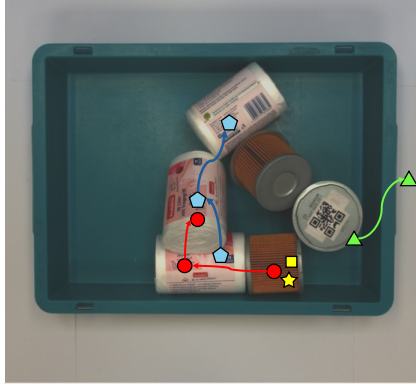
Which object must be removed first to grasp orange propeller?

<think>
orange propeller at (520, 300) is obstructed by bottle cap at (542, 441); bottle cap at (542, 441) is not obstructed.
</think>
<answer><points 542 441>bottle cap</points></answer>

<think>
orange propeller at (300, 470) is not obstructed.
</think>
<answer>{{{"point": [300, 470], "label": "orange propeller"}}</answer>

<think>
orange propeller at (531, 294) is obstructed by blue lid with cat face at (476, 408) with the occlusion ratio of 26%. blue lid with cat face at (476, 408) is obstructed by orange pump bottle at (391, 558) with the occlusion ratio of 11%. orange pump bottle at (391, 558) is not obstructed.
</think>
<answer><points 391 558>orange pump bottle</points></answer>

Figure M. Medium cases on the UNOBench real set.



What is the top object that occludes bottom right cylindrical filter?



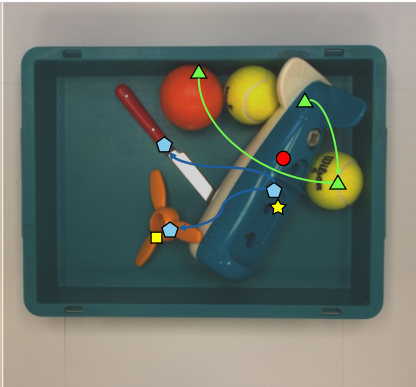
<think>
bottom right cylindrical filter at (1056, 746) is obstructed by top right can at (1322, 540); top right can at (1322, 540) is not obstructed.
</think>
<answer><points 1322 540>top right can</points></answer>



<think>
bottom right cylindrical filter at (670, 500) is obstructed by middle left roll at (530, 470); middle left roll at (530, 470) is obstructed by top left roll at (300, 520); top left roll at (300, 520) is not obstructed.
</think>
<answer>{("point": [300, 520], "label": "top left roll")}</answer>



<think>
bottom right cylindrical filter at (798, 834) is obstructed by bottom white bag roll at (540, 826) with the occlusion ratio of 1%. bottom white bag roll at (540, 826) is obstructed by left white bag roll at (549, 690) with the occlusion ratio of 12%. left white bag roll at (549, 690) is not obstructed.
</think>
<answer><points 549 690>left white bag roll</points></answer>



Tell me the top-most object that blocks access to blue and white handheld device.



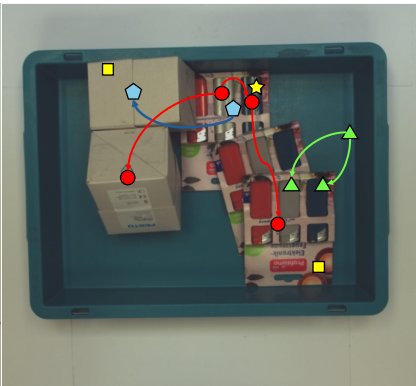
<think>Path1: top-most object at (1032, 574) is obstructed by orange ball at (582, 200); orange ball at (582, 200) is not obstructed. Path2: top-most object at (1032, 574) is obstructed by tennis ball at (888, 300); tennis ball at (888, 300) is not obstructed. Path3: top-most object at (1032, 574) is obstructed by orange ball at (582, 200); orange ball at (582, 200) is not obstructed. Path4: top-most object at (1032, 574) is obstructed by tennis ball at (888, 300); tennis ball at (888, 300) is not obstructed.
</think>
<answer><points 582 200>orange ball</points>, <points 888 300>tennis ball</points></answer>



<think>
Path1: blue and white handheld device at (450, 600) is obstructed by red-handled knife at (368, 430); red-handled knife at (368, 430) is not obstructed. Path2: blue and white handheld device at (450, 600) is obstructed by orange propeller at (600, 420); orange propeller at (600, 420) is not obstructed.
</think>
<answer>{("point": [368, 430], "label": "red-handled knife"), ("point": [600, 420], "label": "orange propeller")}</answer>



<think>
blue and white handheld device at (865, 439) is not obstructed.
</think>
<answer><points 865 439>blue and white handheld device</points></answer>



I want to grasp top package with lighters. What should I remove first?



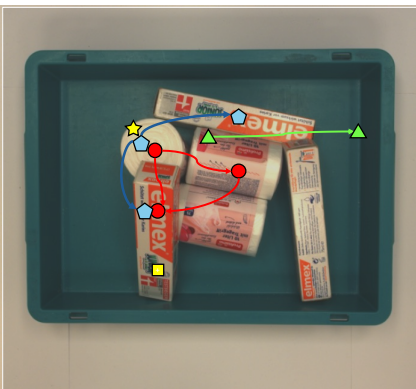
<think>Path1: top package with lighters at (1029, 407) is obstructed by top red lighter at (875, 574); top red lighter at (875, 574) is not obstructed. Path2: top package with lighters at (1029, 407) is obstructed by top white lighter at (972, 574); top white lighter at (972, 574) is not obstructed.
</think>
<answer><points 1029 407>top package with lighters</points>, <points 875 574>top red lighter</points>, <points 972 574>top white lighter</points></answer>



<think>
top package with lighters at (254, 546) is obstructed by top left box at (230, 390); top left box at (230, 390) is not obstructed.
</think>
<answer>{("point": [230, 390], "label": "top left box")}</answer>



<think>Path1: top package with lighters at (648, 279) is obstructed by bottom left box at (375, 553) with the occlusion ratio of 1%. bottom left box at (375, 553) is not obstructed. Path2: top package with lighters at (648, 279) is obstructed by top lighter package at (740, 282) with the occlusion ratio of 1%. top lighter package at (740, 282) is obstructed by bottom right package at (831, 693) with the occlusion ratio of 43%. bottom right package at (831, 693) is not obstructed.
</think>
<answer><points 375 553>bottom left box</points>, <points 831 693>bottom right package</points></answer>



Tell me the top-most object that blocks access to top bandage roll.



<think>top bandage roll at (580, 410) is obstructed by top box at (1049, 410); top box at (1049, 410) is not obstructed.
</think>
<answer><points 1049 410>top box</points></answer>



<think>Path1: top bandage roll at (360, 400) is obstructed by top left box at (490, 410); top left box at (490, 410) is not obstructed. Path2: top bandage roll at (360, 400) is obstructed by top right box at (300, 560); top right box at (300, 560) is not obstructed.
</think>
<answer>{("point": [490, 410], "label": "top left box"), ("point": [300, 560], "label": "top right box")}</answer>



<think>
Path1: top bandage roll at (428, 420) is obstructed by left toothpaste box at (445, 603) with the occlusion ratio of 17%. left toothpaste box at (445, 603) is not obstructed. Path2: top bandage roll at (428, 420) is obstructed by middle bandage roll at (697, 523) with the occlusion ratio of 1%. middle bandage roll at (697, 523) is obstructed by left toothpaste box at (445, 603) with the occlusion ratio of 12%. left toothpaste box at (445, 603) is not obstructed.
</think>
<answer><points 445 603>left toothpaste box</points></answer>

Figure N. Real set failed cases.



Figure O. **Object set used in Real-Robot experiments.** The set comprises 25 distinct household objects varying in shape, size, texture, and deformability. These objects were arranged to create 30 unique scenarios ranging from Easy (shallow piles) to Hard (deep occlusion chains).