RoboMP^2: A Robotic Multimodal Perception-Planning Framework with Multimodal Large Language Models

Qi Lv [1 2] **Hao Li** [1] **Xiang Deng** [1] **Rui Shao** [1] **Michael Yu Wang** [2] **Liqiang Nie** [1]

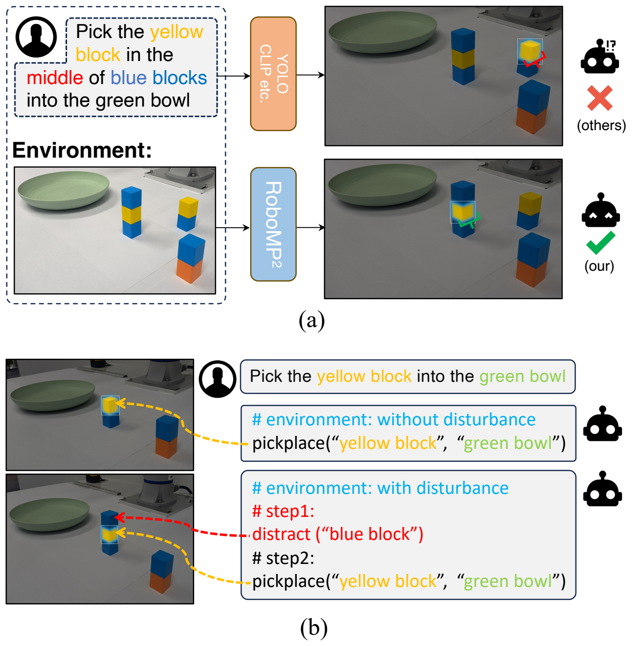
RoboMP2.github.io

Abstract

Multimodal Large Language Models (MLLMs) have shown impressive reasoning abilities and general intelligence in various domains. It inspires researchers to train end-to-end MLLMs or utilize large models to generate policies with human-selected prompts for embodied agents. However, these methods exhibit limited generalization capabilities on unseen tasks or scenarios, and overlook the multimodal environment information which is critical for robots to make decisions. In this paper, we introduce a novel **Robo**tic **M**ultimodal **P**erception-**P**lanning (**RoboMP^2**) framework for robotic manipulation which consists of a Goal-Conditioned Multimodal Preceptor (GCMP) and a Retrieval-Augmented Multimodal Planner (RAMP). Specially, GCMP captures environment states by employing a tailored MLLMs for embodied agents with the abilities of semantic reasoning and localization. RAMP utilizes coarse-to-fine retrieval method to find the k most-relevant policies as in-context demonstrations to enhance the planner. Extensive experiments demonstrate the superiority of RoboMP^2 on both VIMA benchmark and real-world tasks, with around 10% improvement over the baselines.

1. Introduction

Multimodal Large Language Models (MLLMs) (GPT, 2023; Liu et al., 2023) have exhibited remarkable intelligence in various vision and language tasks. Inspired by the impressive performance, many efforts (Ahn et al., 2022; Huang et al., 2022) have been made on using large models to empower embodied agents with human-like intelligence. These


Pick the yellow block in the middle of blue blocks into the green bowl
Environment:
YOLO CLIP etc.
RoboMP2
(others)
(our)
(a)
Pick the yellow block into the green bowl
(b)

```
# environment: without disturbance
pickplace("yellow block", "green bowl")
```

```
# environment: with disturbance
# step1:
distract ("blue block")
# step2:
pickplace("yellow block", "green bowl")
```

Figure 1: (a) The detection results of the yellow block with the complex spatial reference using different methods; (b) Different plans for different environment, even if the same instruction.

approaches mainly use large models to train end-to-end policies (Driess et al., 2023; Brohan et al., 2023b;a) or generate plans with human-selected prompts (Liang et al., 2023; Huang et al., 2023b). However, end-to-end policies and human-selected prompts lack generalization and flexibility on unseen tasks.

Environment perception and task planning are two fundamental components for embodied agents to complete a task (Ahn et al., 2022; Sarch et al., 2023). Most of the existing work typically employs mainstream vision models as environment perceptors, such as YOLOv5 (Yue et al., 2022) and CLIP (Radford et al., 2021)). These models work well in simple scenarios where the categories of the objects are pre-defined or the relationships among objects are easy to be captured. However, they lack the capability to identify and locate objects in unseen scenarios or objects with intricate spatial relationships. For instance, as illustrated in *Fig-*

[1]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen [2]Great Bay University. Correspondence to: Qi Lv <23b951033@stu.hit.edu.cn>.

ure 1(a), the task entails picking up the "yellow block in the middle of the blue blocks". The existing perceptors struggle to accurately identify and locate the specified yellow block since these models cannot understand the semantic information of the complex referring expression. Therefore, it necessitates the development of robot perceptors with multimodal understanding and reasoning capabilities. In light of this, we turn to MLLMs as environment perceptors since they have shown remarkable semantic understanding and vision perception abilities in various tasks (Liu et al., 2023). We accordingly propose a MLLM-based Goal-Conditioned Multimodal Perceptor (GCMP) with an advanced ability to detect objects with a given semantic-complex reference goal. As shown in *Figure* 1(a), GCMP enables robot agents to accurately identify and locate the referred object while the perceptors adopted in the existing embodied frameworks fail.

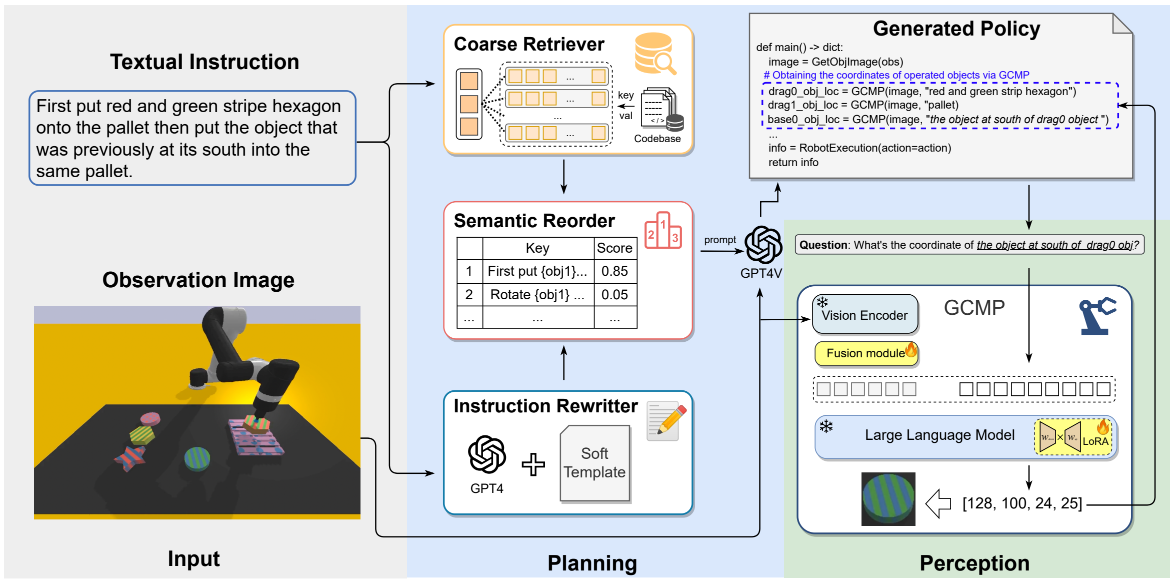
In addition to the multimodal environment perceptor, the planning for subsequent execution is also critical. The existing policies mainly include end-to-end models (Brohan et al., 2023b) and prompt-based approaches (Huang et al., 2023b). The end-to-end policy integrates the perception and planning into a single model, thus requiring closed-loop robot data. However, in the real world, the closed-loop data is very limited due to the expensive human labor for collection. Consequently, these models are proven to overfitting (Ahn et al., 2022; Jiang et al., 2023) the data-collection scenario and show limited generalization in unseen environments or on new tasks. On the other hand, current prompt-based methods rely on manually designed and selected prompt templates to prompt LLMs to generate plans for a given task. They inherently lack generalization (Arenas et al., 2023; Huang et al., 2023a) in diversities of tasks that are highly different from the tasks given in the demonstrations of the prompt templates. To cover different types of tasks, these methods adopt large amounts of templates, which results in in-context attention distraction. Liu et al. (2021) have also demonstrated that redundant in-context examples fail to offer effective guidance, which is detrimental to the ability of the model (Liu et al., 2021). More importantly, these approaches (Huang et al., 2022; Vemprala et al., 2023) typically generate plans based solely on the text instruction, overlooking critical environmental information. For example, as shown in *Figure* 1(b), even for the same instruction, different environments lead to different plans. To address these issues, we propose a retrieval-augmented method which prompts MLLMs to generate plans via adaptively choosing the most relevant policies as demonstrations.

In this paper, we aim to explore how to sufficiently leverage both multimodal information in an environment and the general intelligence in large models to enhance the perception and reasoning capabilities of embodied robots. We propose a novel **Robo**tic **M**ultimodal **P**erception-**P**lanning framework (**RoboMP^2**) which is composed of a Goal-Conditioned Multimodal Perceptor (GCMP) and a Retrieval-Augmented Multimodal Planner (RAMP). Specifically, to endow the embodied model with semantic reasoning and localization capabilities, we propose GCMP to comprehensively perceive environmental information through the integration of a tailored MLLM. Meanwhile, to enhance the generalization of policy planning, we devise RAMP to adaptively find the k most-relevant policies as in-context demonstrations with dedicated coarse retriever, fine reranker. Our main contributions are as followed:

- Different from the existing robot perceptors that can only identify objects with pre-defined classes or simple references, we introduce a taiored MLLM as the environment perceptor, i.e., GCMP, which owns the comprehension abilities to perceive targeted objects with complex references.
- Different from the existing code planners that simply generate code based solely on a text instruction with manually selected templates, we propose RAMP that integrates multimodal environment information into the code generation process, and develops a retrieval-augment strategy to mitigate the interference of redundant in-context examples.
- Extensive experiments show that our proposed RoboMP^2, which is composed of GCMP and RAMP, outperforms the baselines by around 10% on both VIMABench and real-world tasks.

2. Related Work

MLLMs for Robotic Manipulation. Robotic manipulation aims to complete a specific task by interacting with the environment. In recent, imitation learning (Brohan et al., 2023b;a; Chi et al., 2023) has achieved a great success in robotic learning. However, due to the task complexity, it needs large amounts of data to train a robot agent to achieve strong generalization capability (Fang et al., 2023a; Vuong et al., 2023; Shridhar et al., 2022), which is time-consuming and computational-unfriendly. Therefore, many efforts (Huang et al., 2023a;b; Xu et al., 2023) have been made on prompting LLMs to generate policies in a zero-shot manner to control robot agent. Liang et al. (2023) proposed using programs as the policy to control robots by generating calls to perception APIs and control APIs. It has been proven that the prompt is crucial when using LLMs to generate text in a zero-shot manner (Liu et al., 2021). Thus, many approaches (Arenas et al., 2023; Vemprala et al., 2023; Singh et al., 2023) explored the construction of the text prompt, including descriptions of the task details, robotic API libraries, environment feedback (Sarch



```
def main() -> dict:
    image = GetObjImage(obs)
  # Obtaining the coordinates of operated objects via GCMP
    drag0_obj_loc = GCMP(image, "red and green strip hexagon")
    drag1_obj_loc = GCMP(image, "pallet)
    base0_obj_loc = GCMP(image, "the object at south of drag0 object ")
    ...
    info = RobotExecution(action=action)
    return info
```

	Key	Score
1	First put {obj1}...	0.85
2	Rotate {obj1} ...	0.05
...	...	...

Figure 2: Overview of our proposed **RoboMP^2** framework. The three parts in grey/blue/green represent the input data, planning and perception, respectively. The modules highlighted are trainable, including fusion module and LoRA.

et al., 2023). One constraint of previous studies is that they merely utilized the textual information to prompt LLMs, ignoring the significant multimodal information of the environment. Moreover, it would confuse the model if there are too many prompt templates in-context examples, resulting in a wrong execution plan. Thus, in order to alleviate these problems, we introduce a retrieval-augmented multimodal planner which sufficiently leverages the multimodal information and most relevant demonstrations for task planning.

Retrieval Augmented Generation with MLLMs. Retrieval augmented generation (RAG) was first introduced to serve as more informative inputs to unlesh the extensive knowledge of LLMs (Lewis et al., 2021; Liu et al., 2021). Due to its effectiveness, it was subsequently introduced to the multimodal domain. It assists models generate answers by retrieving contents related to the original input as supplement context. RETRO (Borgeaud et al., 2022) introduced additional retrieval encoders to train a GPT model from scratch (Radford et al., 2019). Atlas (Izacard et al., 2023) incorporated similar retrieval encoders to continually finetune T5 model (Raffel et al., 2020). In addition, KNN-based retrieval methods (Fan et al., 2021) and cross modal semantic-based retrieval methods (Zhou et al., 2022) are commonly used to gather information from different modalities, aiming to generate more satisfied text by providing evidence. Considering the plans for robotic manipulation tasks are predominantly involved in the executed actions and target objects, we incorporate a task rewriting module. This module is introduced to extract essential textual information from task instructions. By integrating this module with a coarse-to-fine retrieval method, we adaptively select policies that exhibit semantic similarity as demonstrations, aiming to unveil the inherent capabilities of MLLMs.

3. Framework

In this section, we present a robotic multimodal perception-planning framework (RoboMP^2) that utilizes a Goal-Conditioned Multimodal Perceptor (GCMP) and a Retrieval-Augmented Multimodal Planner (RAMP) to enhance the perception and planning abilities of embdied agents. The framework of RoboMP^2 is shown in *Figure* 2.

3.1. Goal-Conditioned Multimodal Perceptor

3.1.1. Environment Perception for Manipulation

Robotic manipulation involves various types of tasks, imposing different requirements on the perceptor of the robot. For example, when a task entails picking up “an apple”, the robot perceptor needs to have the ability to identify and locate the apple for robot execution. While the perceptors in the existing studies can work well in recognizing this kind of simply referred objects, e.g., pre-defined classes of objects, they struggle to identify semantic-complex referred objects. For instance, when the task involves picking up “the two green apples on the left of the yellow cup”, the existing perceptors fail, since they cannot understand the instruction semantics or the spatial relationships among different objects. In practice, these complex references about objects widely exist in different tasks. We provide three types of commonly used reference expressions that can hardly be handled by the existing perceptors, i.e., referring objects based on attributes, spatial relationships, and knowledge reasoning:

(1) Object Perception Based on Attributes: Many tasks require robots to manipulate objects with specific attributes. As shown in *Figure* 3(b), the robot is asked to manipulate “all the objects with the same color of the blue cube”. It involves one-to-many perception, where a single attribute pertains to many objects. This kind of manipulation requires the perceptor to have a powerful attribute-aware perception ability for not leaving out any of the referred objects.

(2) Object Perception Based on Spatial Relationships: Robots often receive instructions to manipulate objects at specific positions. For example, in *Figure* 3(c), there are two orange blocks, and the instruction is to manipulate “the orange block at the bottom of the purple block”. This requires the perceptor to have spatial-aware perception capabilities.

(3) Object Perception Based on Knowledge Reasoning: Robot manipulation also face situations where knowledge reasoning is needed to determine the operational target based on instructions. As illustrated in *Figure* 3(d), supposing the television stops working, people ask the robot for handing him an object capable of repairing the television. It requires the robot to have high-level understanding and reasoning abilities.

The existing studies (Liang et al., 2023; Huang et al., 2023a) which employ Yolov5 or CLIP can locate objects with the basic object name or simple semantics, such as the example in *Figure* 3(a). However, they lack the ability of perceiving objects with complex referential expressions, such as the the above three kinds of expressions. To address this issue, we built a Goal-Conditioned Multimodal Perceptor (GCMP) upon a MLLM that have both semantic understanding and vision perception abilities.

3.1.2. Training of Multimodal Perceptor

To learn a tailored multimodal perceptor for embodied agents, we formulate robotic data into instruction-tuning format, namely {`image`, `ref_exp`, `coordinates`} triplets, where the input consists of an image and a referential expression (ref_exp), and the output is the corresponding coordinates for manipulation. An example is shown in *Figure* 4. For model architecture, we adopt the ViT (Dosovitskiy et al., 2020) and flan-t5-xl (Chung et al., 2022) as our embodied vision encoder and language encoder. Then, we connect these two encoders through a MLP layer. In addition, we introduce a LoRA module (Hu et al., 2021) to tune the LLM efficiently and effectively.

During instruction tuning, we employ the natural language formulation of all question-answer pairs as the input $(\mathbf{x}^{\mathrm{I}}, \mathbf{x}^{\mathrm{T}})$ and output $\mathbf{y}^{\mathrm{T}}$. The answer is also uniformly formulated as natural language and we adopt the autoregressive training paradigm to tune the model. The learning objective is as followed:

$$\mathcal{L} = -\sum_{k} \log P(y^{\mathrm{T}}_{k+1} | y^{\mathrm{T}}_{i<k}, (\mathbf{x}^{\mathrm{I}}, \mathbf{x}^{\mathrm{T}}); \theta), \qquad (1)$$

where the θ denotes the learnable weights of the model and $\mathbf{x}^{\mathrm{I}}$ denotes the environment image.

3.2. Retrieval-Augmented Multimodal Planner

Planning is critical for embodied agents to complete tasks. Exiting approaches typically prompt LLMs to generate plans according to a textual task instruction and a manually selected prompt template. However, they suffer from two issues: (1) utilizing human selected templates for a given instruction while hardly generalizing to new tasks; (2) only using text information while ignoring multimodal environment information. To address these challenges, we introduce RAMP which is composed of a coarse retriever, a instruction and a fine reranker to adaptively find the k most-relevant policies as in-context demonstrations, thereby boosting the generalization of the policy planning.

3.2.1. Coarse Retriever

To cover diversities of tasks, a prompt typically includes many in-context examples as demonstrations to prompt LLMs to generate task plans. However, this also leads to attention diffusion across these examples, which hurts the quality of the generated plan. To address this issue, we propose a coarse retriever to find the most relevant policies from a codebase. Specifically, we initially assemble a codebase comprising over 50 programs gathered from 10 diverse sources (Arenas et al., 2023; Xu et al., 2023; Liang et al., 2023). Given the textual query $\mathbf{x}^{\mathrm{que}}$ of a task instruction, the coarse retriever aims to recall K programs $\{\mathbf{x}^{\mathcal{C}}_i\}^{K}_{i=1}$ from codebase $\mathcal{C}$, where N represents the number of target candidates. We adopt the two-tower architecture $f(\cdot)$ which extracts the information of $\mathbf{x}^{\mathrm{que}}$ and $\mathbf{x}^{\mathcal{C}}_i$ separately. The score between task instruction query $\mathbf{x}^{\mathrm{que}}$ and candidate $\mathbf{x}^{\mathcal{C}}_i$ is as followed:

$$\mathbf{V}^{\mathrm{que}} = f(\mathbf{x}^{\mathrm{que}}),\ \ \mathbf{V}^{\mathcal{C}}_i = f(\mathbf{x}^{\mathcal{C}}_i), \qquad (2)$$

$$\mathrm{Score}^{\mathrm{CR}}_i = \mathbf{V}^{\mathrm{que}} \circ \mathbf{V}^{\mathcal{C}}_i, \qquad (3)$$

where $\circ$ represents the dot product operation.

We adopt TF-IDF (Aizawa, 2003) to compute the similarity. Meanwhile, extensive experiments are conducted to compare this metric and other two classic metrics, i.e., BM25 (Robertson et al., 2009) and SentenceBERT (Reimers & Gurevych, 2019). The details of similarity computations and the experimental comparison can be seen in Appendix A and Section 4.5.

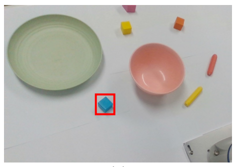
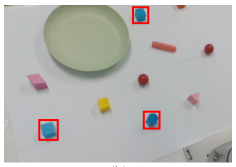
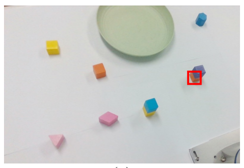
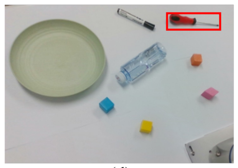
Type	Object Name	Attribute	Spatial Relationship	Knowledge Reasoning
Manipulated Object	the blue cube	all objects with the same color of the blue cube	the orange cube at the bottom of the purple cube	the object which has ability to repair a TV
Location	[136, 167, 164, 138]	[51, 205, 81, 175] [195, 28, 218, 6] [212, 188, 236, 163]	[232, 153, 286, 139]	[242, 37, 323, 15]
Visualization	 (a)	 (b)	 (c)	 (d)

Figure 3: Examples of manipulated objects with four different referential relationships types.

```
Ref_exp: the coordinate of the object to the
right of the green plate, second from the right
Coordinates: [300, 66, 324, 88]
```

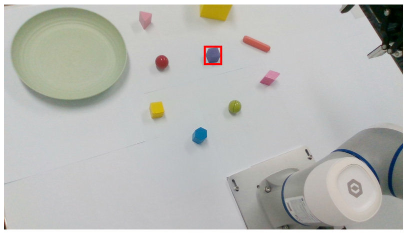


Figure 4: An example of training data for the GCMP.

3.2.2. Fine Reranker

Despite relevant code snippets could be recalled through the coarse retriever, there remains an issue of the order among candidates. In the planning generation phase, large models are highly sensitive to the sequence of demonstrations. Consequently, we introduce a rewriting module and a reranker module respectively to extract its core of the task instruction and order these relevant demonstrations. Finally, we only use the k most relevant programs as the in-context examples.

Instruction Rewriter. The robotic policy is mainly related to the action type and manipulation objects. Thus, we introduce a instruction rewriting module to eliminate distracting expressions from the task description, obtaining its core robotic operation instructions. Specifically, we combine the raw textual task query $\mathbf{x}^{\text{que}}$ and a soft prompt template to rewrite $\hat{\mathbf{x}}^{\text{que}}$. The formula is as followed:

$$\hat{\mathbf{x}}^{\text{que}} = \text{Rewritter}(\mathbf{x}^{\text{que}}, T_{\text{soft}}), \tag{4}$$

where we use the GPT4 as the rewritter and T_{soft} denotes the soft prompt template which consists of `Scene`, `Description`, `Example` and `Instruction`. The first two primarily provide premises and requirements for the robot scenarios, while the latter two consist of concrete examples and the corresponding task instructions. More details can be seen in Appendix B.

Semantic Reorder. In the plan generation stage, the order of the examples is crucial. Due to the design of positional encoding in the transformer architecture, researchers (Li et al., 2023; Rubin et al., 2021) have found that model generation tends to primarily focus on the content at the beginning and the end of the paragraph. Therefore, we introduce a reorder module with the aim of sorting the retrieved candidate code snippets$\{\mathbf{x}_i^{\mathcal{C}}\}_{i=1}^{K}$. A straightforward solution is to simply concatenate it at the beginning of the whole candidate.

$$\mathbf{H}_i = \text{SentenceBERT}([\hat{\mathbf{x}}^{\text{que}}, \mathbf{x}_i^{\mathcal{C}}]), \tag{5}$$

$$\text{Score}_i^{\text{FR}} = \text{Sigmoid}(\mathbf{H}_i). \tag{6}$$

In the end, the coarse-grained ranking scores and fine-grained ranking scores would be applied for the comprehensive ranking as followed:

$$\text{Score}_i = \lambda \cdot \text{Score}_i^{\text{CR}} + (1-\lambda) \cdot \text{Score}_i^{\text{FR}}, \tag{7}$$

$$\propto p(x_{[\text{cls}]} | \mathbf{x}^{\text{que}}, \hat{\mathbf{x}}^{\text{que}}, \mathbf{x}_i^{\mathcal{C}}). \tag{8}$$

Equation 8 illustrates the relationship between the Score_i and $(\mathbf{x}^{\mathcal{C}}, \mathbf{x}_i^{\text{que}}, \hat{\mathbf{x}}_i^{\text{que}})$, when SentenceBERT is employed for both the coarse retriever and the fine reranker.

3.2.3. Multimodal Generation Module

After adaptively selecting k most relevant examples $\{\mathbf{x}_i^{\mathcal{C}}\}_{i=1}^{k}$ through the coarse-to-fine retrieval method, we combine them with a template to construct the complete prompt, including the third-party libraries, function definitions, and the task instruction. It should be noted that we arrange the examples in reverse order based on relevance, meaning the most-relevant examples are placed towards the end. Compared with the pure textual generator, the multimodal information of current environment is crucial. Hence

	L1	L2	L3	L4	Avg.
End-to-end models					
Gato[†]	58.1	53.2	43.5	12.4	41.8
Flamingo[†]	47.5	46.0	40.8	12.1	36.6
VIMA[†]	81.6	81.5	79.0	48.9	72.7
RT-2	72.8	70.3	66.8	47.0	64.2
MLLM Planners					
CaP	71.2	70.0	42.8	44.7	57.2
VisualProg	49.7	47.7	69.9	52.2	54.9
I2A[†]	77.0	76.2	66.6	49.8	65.0
RoboMP^2	**89.0** (+7.4)	**85.9** (+4.4)	**86.8** (+7.8)	**68.0** (+19.1)	**82.4** (+9.7)

Table 1: The results on the VIMA Benchmark. [†] denotes the cited result.

Task Name	I2A	RoboMP^2
Basic Task		
Visual Manipulation	85.0	90.0
Same Shape	45.0	90.0
Same Color	40.0	90.0
Challenging Task		
Manipulate Old Neighbor	0.0	70.0
Pick in Order then Restore	65.0	80.0
Interfering Manipulation	0.0	55.0
Avg. Success Ratio (%)	39.2	**79.2**

Table 2: The results on the real-world tasks.

we use the GPT4V as the multimodal generator with the input $\{\mathbf{x}^{\mathrm{que}}, \mathbf{x}^{\mathrm{I}}, \mathbf{x}_i^{\mathcal{C}}\}_{i=1}^{k}$ and full prompt template. Examples of full template are presented in Appendix D.

4. Experiment

4.1. Data and Evaluation Metrics

We employ VIMA (Jiang et al., 2023) as the test benchmark which encompasses 17 tasks ranging from L1-level to L4-level difficulty. In the zero-shot real-world experiment, we construct 7 tasks including 4 sim-to-real tasks and 3 generalization tasks. We measure the performance of different methods through the Success Rate (SR) metrics. Please refer to Appendix C for more details.

4.2. Baselines

We compare our method with the following two types of baselines:

End-to-end models: (1) **Gato** (Reed et al., 2022) introduces a decoder-only architecture which is prompted with the observation and action subsequence; (2) **Flamingo** (Alayrac et al., 2022) embeds a variable number of prompt images into a fixed number of tokens as Perceiver and an additional robot action heads; (3) **VIMA** (Jiang et al., 2023) leverages the multimodal prompt and x-attention block to fuse the instruction and observation information; (4) **RT-2** (Jiang et al., 2023) trains a large robotic model by stacking multi-transformer decoder blocks via autoregressive generation.

Prompt-based methods: (5) **VisProg** (Huang et al., 2023a) generates python-like modular programs, which are then executed to get both the solution and a comprehensive and interpretable rationale. (6) **CaP** (Liang et al., 2023) generates the code as policies to manipulate the robot given the textual instruction. (7) **I2A** (Huang et al., 2023a) follows the code policy style while introducing the off-the-shelf SAM (Kirillov et al., 2023) and CLIP (Radford et al., 2021) to obtain coordinates of target objects.

4.3. Implementation Details

For GCMP in our proposed RoboMP^2, we adopt the `EVA-CLIP/g` (Fang et al., 2023b) as the visual encoder of our perceptor. Regarding the LLM, we investigate the encoder-decoder architecture model `flan-t5-xl` (Chung et al., 2022). During training, the vision encoder and the LLM are frozen, only learning the weight of the fusion module and the LoRA module. We set the epoch to 10, the batch size to 128, the learning rates of the fusion module and LoRA module to 3e-5 and 1e-4, respectively. We adopt the AdamW optimizer and the cosine decay learning schedule. The overall training time is around 24 hours on a 8*A100-80G-SXM4 platform.

For RAMP in the proposed RoboMP^2, we adopt the GPT4/GPT3.5 and GPT4V as the text-only rewritter and the multimodal generator. The number of the most-relevant examples during coarse retriever and fine reranker is set to 5 and 2, respectively.

4.4. Results

Experimental Results on VIMABench. The task difficulty of VIMABench ranges from L1 level to L4 level, with tasks at L1-L3 levels being seen, while tasks at L4 level are unseen. As shown in *Table* 1, our RoboMP^2 outperforms other methods by a large extent, surpassing the VIMA baseline around 10% on the average performance of success ratio. This demonstrates the effectiveness of RoboMP^2 for robotic manipulation.

Compared with the other baselines, the average performance improvements of RoboMP^2 range from 4% to 8% on seen tasks (L1-L3), while a comprehensive improvement of approximately 20% on unseen tasks (L4). The reason is that end-to-end models tend to overfit to the training tasks due to the small amount of closed-loop data, resulting in limited generalization abilities on unseen tasks. The other prompted-based methods use fixed human-selected in-context examples in the prompt, thus lacking adaptability on diverse unseen tasks. In contrast, the proposed RoboMP^2 shows a much better generalization ability on unseen tasks since RAMP in RoboMP^2 can adaptively find the most relevant policies as their demonstrations to prompt the generator according to the task.

Real-world Experimental Results. We conduct experiments on 6 real-world tasks, comparing the performance of RoboMP^2 with that of I2A. This is due to that both of them can be transferred from simulation to reality, without requiring to collect data for additional training. As illustrated in *Table* 2, our RoboMP^2 outperforms I2A by 40% in terms of the overall average success ratio. It can be seen that I2A performs well in the task `visual manipulation`. However, it struggles to complete the manipulation tasks of objects with specific attributes such as `same shape`, let alone challenging tasks. In comparison, RoboMP^2 shows consistent improvements across these two tasks.

It is noted that, on the tasks requiring to recognize objects with the same attribute or a specific relationship such as `manipulate old neighbor`, the performance of I2A drops significantly, even to zero. The reason is that the SAM+CLIP perceptor in I2A is unable to simultaneously identify multiple instances and is hardly handle the detection of objects with complex relationships. Benefiting from our proposed Goal-Conditioned Multimodal Perceptor (GCMP), RoboMP^2 demonstrates a strong capability in dealing with such challenges.

4.5. Ablation Study

Effects of the Coarse-to-fine Retriever. We conduct ablation experiments to valid the performance of RoboMP^2 with different settings of the retriever-augmented module. In particular, we compare 4 different implementations: (1) *w/o CR* removes the coarse retriever, directly using the rewriting module to order all programs in the codebase; (2) *w/o IR* removes the rewriting module, remaining the coarse retriever and semantic reorder; (3) *w/o SR* removes the reorder module, unsorting the retrieved demonstrations; (4) *w/o RAMP* removes the whole retriever, and takes k programs from the codebase randomly as prompt examples. As depicted in *Table* 4, without either of these components, the average performance drops significantly. The reason is that when the context is composed of irrelevant examples, it is difficult for the planner to generate execution plans successfully.

In addition, λ in RAMP balances the contributions between the similarity score of the coarse retriever and the fine reranker. We investigate its effects with different values ranging from 0 to 1, as well as different similarity calculation methods for the coarse retriever, including TF-IDF, BM25, and SentenceBERT. As illustrated in *Table* 4, it achieves the best performance when λ is set to 0.25 and TF-IDF performs consistently better than the other similarity calculation methods for the coarse retreiver. Since the input for the fine reranker is derived from candidates provided by the coarse retriever, the distinctions among these candidates are minimal. Consequently, it is essential to assign a higher weight to the fine reranker to ensure its significant impact during the reranking process. On the other hand, instructions for the same task often contain representative keywords on VIMABench. Therefore, the TF-IDF similarity calculation method which is based on keyword frequency matching could achieve excellent results.

To investigate the robustness of RoboMP^2, we construct an extensive test set by enriching the original instructions using GPT4. Subsequently, we conduct experiments to evaluate the impact of this augmentation. As shown in *Table* 3, the results demonstrate the outstanding semantic understanding capability of our approach. In comparison to I2A, RoboMP^2 is not confined to processing text instructions with fixed patterns. On the contrary, it possesses stronger generalization and robustness.

Effectiveness of Multimodal Planner. To assess the influence of multimodal information on plan generation, we replaced the planning generator from GPT4V, which accepts both text and images as the input, with GPT3.5 and GPT4, both of which rely solely on the text input. The results on the VIMA Benchmark are illustrated in *Table* 6. It can be seen that the performance partially decreases when the visual input is absent. Upon analyzing the generated code plans, we found GPT4 fails to detect these anomalies, when there are disturbances in the environment without explicit mention in the task, as illustrated in *Figure* 1(b). In contrast, with the assistance of visual information, GPT4V can effectively perceive environmental factors, enabling it to make correct decisions.

Comparison of Multimodal Perceptors. To validate the capabilities of GCMP for intricate relationship references, we construct a perception test set. This set contains fundamental perception tasks that detect objects by names, and complex perception tasks that requires understanding complex referential expressions as illustrated in Section 3.1.1. Since the traditional visual perceptors used in the robotic manipulation, such as YOLO, cannot take the referential expressions to perceive objects, we compare our embodied perceptor GCMP with SAM+CLIP in I2A, and the general MLLM, i.e., Shikra (Chen et al., 2023). It can be seen from *Table* 7 that our GCMP outperforms the two models on both simple and complex referential perception by a large margin. Neither SAMP+CLIP or Shikra can work on the complex referential perception. The reason is that SAM+CLIP does not have a ability to understand the complex semantic reference and the general model Shikra struggles to generalize to complex embodied perception tasks.

	Visual manipulate	Rotate	Rearrange	Rearrange then restore	Scene understanding	Pick in order then restore	Avg.
I2A	63.5	3.4	0.0	0.0	2.9	8.4	13.0
RoboMP^2	**92.0**	**52.0**	**84.0**	**88.0**	**60.0**	**28.0**	**67.3**

Table 3: The results on the enriched test set.

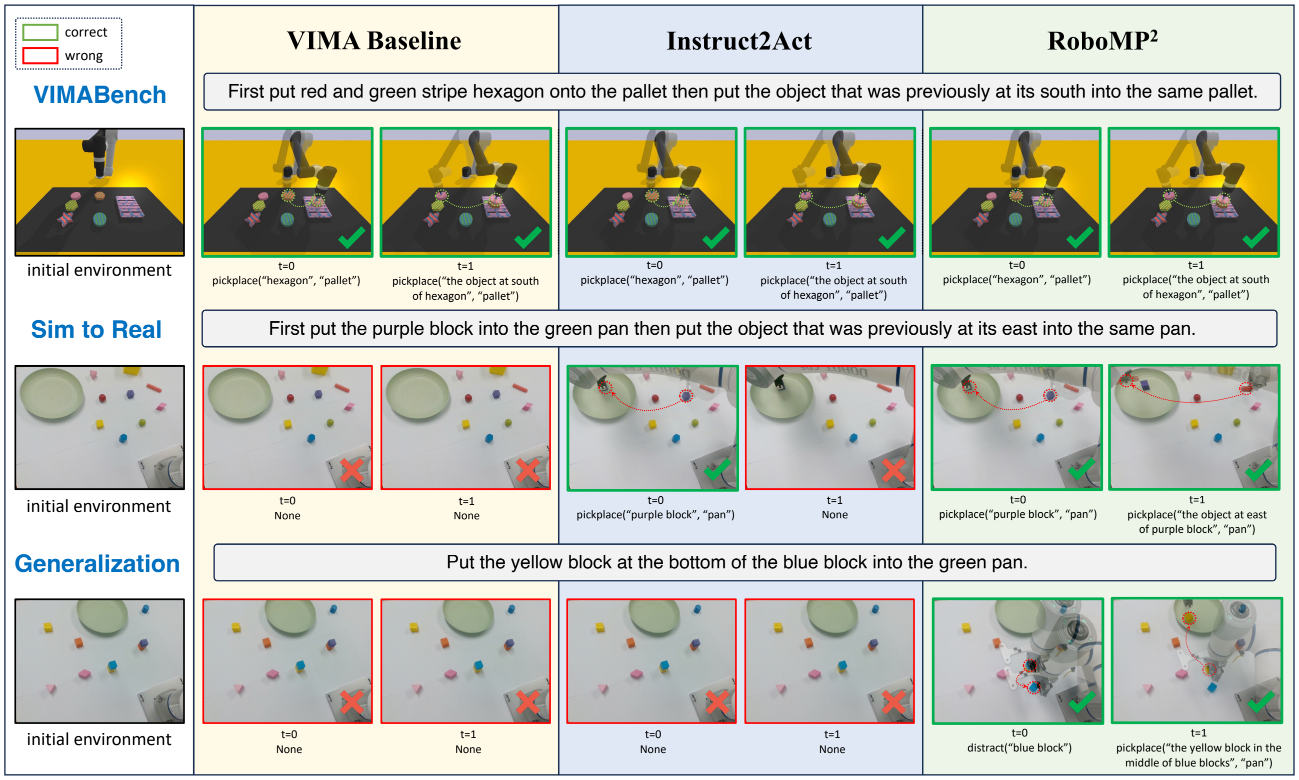

correct
wrong
VIMABench
VIMA Baseline
Instruct2Act
RoboMP²
First put red and green stripe hexagon onto the pallet then put the object that was previously at its south into the same pallet.
initial environment
t=0
pickplace("hexagon", "pallet")
t=1
pickplace("the object at south of hexagon", "pallet")
Sim to Real
First put the purple block into the green pan then put the object that was previously at its east into the same pan.
initial environment
t=0
None
t=1
None
pickplace("purple block", "pan")
pickplace("the object at east of purple block", "pan")
Generalization
Put the yellow block at the bottom of the blue block into the green pan.
initial environment
distract("blue block")
pickplace("the yellow block in the middle of blue blocks", "pan")

Figure 5: Visualization of robot planning and execution on VIMABench, zero-shot sim-to-real and generalization tasks.

	CR	IR	SR	L1	L2	L3	L4	Avg.
RoboMP^2	✓	✓	✓	**89.0**	85.9	**86.8**	**68.0**	**82.4**
RoboMP^2 $_{\text{w/o CR}}$	✗	✓	✓	14.4	16.0	19.6	8.0	14.5
RoboMP^2 $_{\text{w/o IR}}$	✓	✗	✓	87.7	**86.1**	86.7	49.0	77.4
RoboMP^2 $_{\text{w/o SR}}$	✓	✓	✗	81.9	80.6	82.7	65.0	77.5
RoboMP^2 $_{\text{w/o RAMP}}$	✗	✗	✗	10.5	9.5	11.6	3.0	8.7

Table 4: The performance of RoboMP^2 variants with removing different retriever components on VIMABench.

	λ=0	λ=0.25	λ=0.5	λ=0.75	λ=1
TF-IDF	74.1	**82.4**	72.4	72.6	73.9
BM25	42.8	43.1	42.7	42.8	41.8
S-BERT	35.9	33.9	35.7	35.9	34.2

Table 5: The comparison results regarding different values of λ and similarity metrics on the VIMABench dataset.

	Text	Image	L1	L2	L3	L4	Avg.
RoboMP^2 $_{\text{w/ GPT4V}}$	✓	✓	**89.0**	**85.9**	**86.8**	**68.0**	**82.4**
RoboMP^2 $_{\text{w/ GPT4}}$	✓	✗	87.8	79.6	77.8	65.0	77.6
RoboMP^2 $_{\text{w/ GPT3.5}}$	✓	✗	67.2	73.2	70.7	53.9	66.2

Table 6: The performance of multimodal and text-only planners.

	Simple Perception			Complex Perception		
	mAP$_{0.3}$	mAP$_{0.5}$	mAP$_{0.75}$	mAP$_{0.3}$	mAP$_{0.5}$	mAP$_{0.75}$
SAM$_H$+CLIP$_H$	78.9	74.1	64.1	0	0	0
Shikra	63.6	18.2	0	0	0	0
GCMP	**99.1**	**94.0**	**89.2**	**99.0**	**97.7**	**78.0**

Table 7: The performance of multimodal perceptors.

4.6. Qualitative Results

We further present qualitative results in *Figure* 5, illustrating the planning and execution steps of RoboMP^2 in comparison to other methods across three different kinds of tasks. It can be observed that only RoboMP^2 successfully accomplishes all tasks. While all three methods perform well on VIMABench tasks, the end-to-end policy fails to generalize to real-world scenarios due to differences in the robot arm. Furthermore, SAM+CLIP schema is shown to unable to recognize referential objects with complex relationships. This leads I2A to only complete pick-and-place tasks for simple objects, resulting in failure in the second step of sim-to-real. In the generalization scenario, other methods are unable to complete the instruction. Only RoboMP^2, leveraging its robust language understanding and multimodal perception capabilities, successfully identify objects with intricate relationships. It adaptively generates action strategies based on the current environment, ultimately completing the task.

5. Conclusion

In this paper, we have proposed a novel Robotic Perception and Planning framework (RoboMP^2) that consists of the Goal-Conditioned Multimodal Perceptor (GCMP) and the Retrieval-Augmented Multimodal Planner (RAMP). GCMP is introduced to capture multimodal environment information by incorporating a tailored MLLM. RAMP employs a coarse-to-fine retrieval-augmented approach to adaptively select the k most-relevant policies as in-context demonstrations to enhance the generalization. Extensive experiments demonstrate that RoboMP^2 outperforms the baselines by a large margin on both VIMABench and real-world tasks.

6. Impact Statements

Inspired by the great successes of multimodal large language models (MLLMs), we propose RoboMP^2 that leverages both multimodal information in an environment and the general intelligence in MLLMs to enhance the perception and reasoning capabilities of embodied robots. To this end, this technology is expected to advance smart robots that can free human beings from some tedious work. There are many potential societal consequences of developing smart robots, none which we feel must be specifically highlighted here.

References

Openai. gpt-4v(ision) system card, 2023. URL https://cdn.openai.com/papers/GPTVSystemCard.pdf.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

Aizawa, A. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1): 45–65, 2003.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning, 2022.

Arenas, M. G., Xiao, T., Singh, S., Jain, V., Ren, A. Z., Vuong, Q., Varley, J., Herzog, A., Leal, I., Kirmani, S., Sadigh, D., Sindhwani, V., Rao, K., Liang, J., and Zeng, A. How to prompt your robot: A promptbook for manipulation skills with code as policies. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023. URL https://openreview.net/forum?id=T8AiZj1QdN.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. Improving language models by retrieving from trillions of tokens, 2022.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M. S., Salazar, G., Sanketi, P. R., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H. T., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. RT-1: robotics transformer for real-world control at scale. In Bekris, K. E., Hauser, K., Herbert, S. L., and Yu, J. (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023b. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

Chen, K., Zhang, Z., Zeng, W., Zhang, R., Zhu, F., and Zhao, R. Shikra: Unleashing multimodal llm's referential dialogue magic, 2023.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *CoRR*, abs/2303.04137, 2023. doi: 10.48550/ARXIV.2303.04137. URL https://doi.org/10.48550/arXiv.2303.04137.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Narang, S., Mishra, G., Yu, A., Zhao, V. Y., Huang, Y., Dai, A. M., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416, 2022. doi: 10.48550/ARXIV.2210.11416. URL https://doi.org/10.48550/arXiv.2210.11416.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. Palm-e: An embodied multimodal language model, 2023.

Fan, A., Gardent, C., Braud, C., and Bordes, A. Augmenting transformers with knn-based composite memory for dialog. *Trans. Assoc. Comput. Linguistics*, 9: 82–99, 2021. doi: 10.1162/TACL_A_00356. URL https://doi.org/10.1162/tacl_a_00356.

Fang, H., Fang, H., Tang, Z., Liu, J., Wang, J., Zhu, H., and Lu, C. RH20T: A robotic dataset for learning diverse skills in one-shot. *CoRR*, abs/2307.00595, 2023a. doi: 10.48550/ARXIV.2307.00595. URL https://doi.org/10.48550/arXiv.2307.00595.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., and Cao, Y. Eva: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19358–19369, 2023b.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Huang, S., Jiang, Z., Dong, H., Qiao, Y., Gao, P., and Li, H. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *CoRR*, abs/2305.11176, 2023a. doi: 10.48550/ARXIV.2305.11176. URL https://doi.org/10.48550/arXiv.2305.11176.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models, 2022.

Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., and Fei-Fei, L. Voxposer: Composable 3d value maps for robotic manipulation with language models. *CoRR*, abs/2307.05973, 2023b. doi: 10.48550/ARXIV.2307.05973. URL https://doi.org/10.48550/arXiv.2307.05973.

Izacard, G., Lewis, P. S. H., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43, 2023. URL http://jmlr.org/papers/v24/23-0037.html.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W., Dollár, P., and Girshick, R. B. Segment anything. *CoRR*, abs/2304.02643, 2023. doi: 10.48550/ARXIV.2304.02643. URL https://doi.org/10.48550/arXiv.2304.02643.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Li, F., Jiang, Q., Zhang, H., Ren, T., Liu, S., Zou, X., Xu, H.-S., Li, H., yue Li, C., Yang, J., Zhang, L., and Gao, J. Visual in-context prompting. *ArXiv*, abs/2311.13601, 2023. URL https://api.semanticscholar.org/CorpusID:265351501.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 9493–9500. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning, 2023.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. What makes good in-context examples for gpt-3?, 2021.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent, 2022.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Rubin, O., Herzig, J., and Berant, J. Learning to retrieve prompts for in-context learning. *ArXiv*, abs/2112.08633, 2021. URL https://api.semanticscholar.org/CorpusID:245218561.

Sarch, G., Wu, Y., Tarr, M. J., and Fragkiadaki, K. Open-ended instructable embodied agents with memory-augmented large language models, 2023.

Shridhar, M., Manuelli, L., and Fox, D. Perceiver-actor: A multi-task transformer for robotic manipulation. In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799. PMLR, 2022. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 11523–11530. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161317. URL https://doi.org/10.1109/ICRA48891.2023.10161317.

Vemprala, S., Bonatti, R., Bucker, A., and Kapoor, A. Chatgpt for robotics: Design principles and model abilities. *CoRR*, abs/2306.17582, 2023. doi: 10.48550/ARXIV.2306.17582. URL https://doi.org/10.48550/arXiv.2306.17582.

Vuong, Q., Levine, S., Walke, H. R., Pertsch, K., Singh, A., Doshi, R., Xu, C., Luo, J., Tan, L., Shah, D., Finn, C., Du, M., Kim, M. J., Khazatsky, A., Yang, J. H., Zhao, T. Z., Goldberg, K., Hoque, R., Chen, L. Y., Adebola, S., Sukhatme, G. S., Salhotra, G., Dass, S., Pinto, L., Cui, Z. J., Haldar, S., Rai, A., Shafiullah, N. M. M., Zhu, Y., Zhu, Y., Nasiriany, S., Song, S., Chi, C., Pan, C., Burgard, W., Mees, O., Huang, C., Pathak, D., Bahl, S., Mendonca, R., Zhou, G., Srirama, M. K., Dasari, S., Lu, C., Fang, H.-S., Fang, H., Christensen, H. I., Tomizuka, M., Zhan, W., Ding, M., Xu, C., Zhu, X., Tian, R., Lee, Y., Sadigh, D., Cui, Y., Belkhale, S., Sundaresan, P., Darrell, T., Malik, J., Radosavovic, I., Bohg, J., Srinivasan, K., Wang, X., Hansen, N., Wu, Y.-H., Yan, G., Su, H., Gu, J., Li, X., Suenderhauf, N., Rana, K., Burgess-Limerick, B., Ceola, F., Kawaharazuka, K., Kanazawa, N., Matsushima, T., Matsuo, Y., Iwasawa, Y., Furuta, H., Oh, J., Harada, T., Osa, T., Tang, Y., Kroemer, O., Sharma, M., Zhang, K. L., Kim, B., Cho, Y., Han, J., Kim, J., Lim, J. J., Johns, E., Palo, N. D., Stulp, F., Raffin, A., Bustamante, S., Silvério, J., Padalkar, A., Peters, J., Schölkopf, B., Büchler, D., Schneider, J., Guist, S., Wu, J., Tian, S., Shi, H., Li, Y., Wang, Y., Zhang, M., Amor, H. B., Zhou, Y., Majd, K., Ott, L., Schiavi, G., Martín-Martín, R., Shah, R., Bisk, Y., Bingham, J. T., Yu, T., Jain, V., Xiao, T., Hausman, K., Chan, C., Herzog, A., Xu, Z., Kirmani, S., Vanhoucke, V., Julian, R., Lee, L., Ding, T., Chebotar, Y., Tan, J., Liang, J., Mordatch, I., Rao, K., Lu, Y., Gopalakrishnan, K., Welker, S., Joshi, N. J., Devin, C. M., Irpan, A., Moore, S., Wahid, A., Wu, J., Chen, X., Wohlhart, P., Bewley, A., Zhou, W., Leal, I., Kalashnikov, D., Sanketi, P. R., Fu, C., Xu, Y., Xu, S., brian ichter, Hsu, J., Xu, P., Brohan, A., Sermanet, P., Heess, N., Ahn, M., Rafailov, R., Pooley, A., Byrne, K., Davchev, T., Oslund, K., Schaal, S., Jain, A., Go, K., Xia, F., Tompson, J., Armstrong, T., and Driess, D. Open x-embodiment: Robotic learning datasets and RT-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023*, 2023. URL https://openreview.net/forum?id=zraBtFgxT0.

Xu, M., Huang, P., Yu, W., Liu, S., Zhang, X., Niu, Y., Zhang, T., Xia, F., Tan, J., and Zhao, D. Creative robot tool use with large language models. *CoRR*, abs/2310.13065, 2023. doi: 10.48550/ARXIV.2310.13065. URL https://doi.org/10.48550/arXiv.2310.13065.

Yue, X., Li, H., Shimizu, M., Kawamura, S., and Meng, L. Yolo-gd: a deep learning-based object detection algorithm for empty-dish recycling robots. *Machines*, 10(5):294, 2022.

Zhou, M., Luo, G., Rohrbach, A., and Yu, Z. Focus! relevant and sufficient context selection for news image captioning. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6078–6088, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.450. URL https://aclanthology.org/2022.findings-emnlp.450.

A. Similarity Computation Method

(1) **TF-IDF**: It is a numerical statistic to evaluate the importance of a word within a document relative to a collection of corpus. We obtain the term frequency (TF) and the inverse document frequency (IDF), then multiplying them to compute the the TF-IDF score as followed:

$$\mathrm{TF}(t_i, q) = \frac{n_{t_i,q}}{\sum_{k=1}^{l} n_{t_k,q}}, \tag{9}$$

$$\mathrm{IDF}(t_i, \mathcal{C}_j) = \log\left(\frac{|\mathcal{C}|}{n_{t_i,\mathcal{C}_j} + 1}\right), \tag{10}$$

$$\mathrm{Score}_{\mathrm{TF-IDF}}(q, \mathcal{C}_j) = \sum \mathrm{IDF}(t_i, \mathcal{C}_j) \times \mathrm{TF}(t_i, q), \tag{11}$$

where t_i, q, $\mathcal{C}_j$ represents the term word, query, and program in codebase $\mathcal{C}$. $n_{t_i,q}$ means the number of times term t_i appears in query q while $|\mathcal{C}|$ denotes its total number.

(2) **BM25**: This is an extension of TF-IDF, particularly effective in dealing with long documents. It improves the calculation of IDF and simultaneously enhances the weighting of TF, instead of the original TF. The computation procedure is as followed:

$$\mathcal{W}(t_i, q, \mathcal{C}) = \frac{\mathrm{TF}(t_i, q) \cdot (k_1 + 1)}{\mathrm{TF}(t_i, q) + k_1 \cdot (1 - b + b \cdot \frac{|q|}{L_{avg}(\mathcal{C})})}, \tag{12}$$

$$\mathrm{IDF}(t_i, \mathcal{C}_j) = \log(\frac{|\mathcal{C}| - n_{t_i,\mathcal{C}_j} + 0.5}{n_{t_i,\mathcal{C}_j} + 0.5} + 1), \tag{13}$$

$$\mathrm{Score}_{\mathrm{BM25}}(q, \mathcal{C}_j) = \sum \mathrm{IDF}(t_i, \mathcal{C}_j) \cdot \mathcal{W}(t_i, q), \tag{14}$$

where $\mathcal{W}$ means the weight of each term t_i in query q, and L_{avg} denotes the average length of codebase $\mathcal{C}$.

(3) **SentenceBERT**: Different from keyword similarity computation methods, it embeds the sentence to a vector via a pretrain language model, and then use the cosine similarity to compute their scores.

$$h_q = \mathrm{Encoder}(q) \tag{15}$$

$$h_{\mathcal{C}_j} = \mathrm{Encoder}(\mathcal{C}_j) \tag{16}$$

$$\mathrm{Score}_{\mathrm{SBERT}} = h_q \circ h_{\mathcal{C}_j} \tag{17}$$

B. Rewriting Prompt

As shown in *Figure* 6, we construct a rewritter soft template which consists of robotic scene, description, example and instruction. The first two primarily provide premises and requirements for the robot scenarios, while the latter two consist of concrete examples and the corresponding task instruction.

```
Scene: You are a powerful robot who is performing manipulations based on instructions.

Description: Upon receiving a task, your first step is to extract the atomic actions involved in the textual instructions. You need to
proceed step by step, starting with understanding the natural language instructions, then eliminating interference from object names
and other factors, and finally generating atomic actions.

Example:
    Instruction: Put the red swirl block into the purple container.
    Action: Put the {object1} into the {object2}.

Instruction: <Insert your instruction>
```

Figure 6: The soft template of instruction rewritter.

C. Dataset

C.1. VIMABench

VIMABench (Jiang et al., 2022) introduces a comprehensive four-level evaluation protocol designed to progressively increase in difficulty, challenging trained agents. The protocol comprises four levels, each testing different aspects of the agent's capabilities.

- Level 1 (L1) - Placement Generalization: Randomizing the placement of target objects is required in this level, testing the agent's ability to adapt to varying configurations.
- Level 2 (L2) - Combinatorial Generalization: Generating new combinations of target materials and object descriptions challenges the agent's adaptability to diverse scenarios.
- Level 3 (L3) - Novel Object Generalization: Assessing the agent's capacity to generalize to unfamiliar materials and objects, gauging its ability to handle the unknown.
- Level 4 (L4) - Novel Task Generalization: Requiring agents to ground and execute tasks that have not been encountered before, pushing the boundaries of the agent's problem-solving capabilities.

The tasks, presented in the order of their corresponding task index, align with the structure of VIMABench:

1. Visual Manipulation: Select a designed object and place it into a specific container.
2. Scene Understanding: Recognize a specific object in a provided image and place it into a designed container.
3. Rotate: Rotate a specific object by a designated degree.
4. Rearrange: Target objects in the current state into the scene pattern based on a provided image.
5. Rearrange then Restore: Complete the Rearrange task, then restore objects to their initial placement.
6. Novel Adjectives Understanding: Recognize novel adjectives by comparing object images provided in the VIMABench prompt.
7. Novel Nouns: Recognize novel nouns similar to the above.
8. Novel Nouns and Adjectives: Combine novel adjectives and nouns effectively.
9. Novel Concept Understanding: Recognize degrees indicated in the prompt and rotate a specific object accordingly.
10. Follow Motion: Manipulate objects corresponding to given frames in the VIMABench prompt.
11. Stack the Blocks: Similar to the above, involving stacking blocks.
12. Sweep Without Exceeding: Manipulate specific objects without exceeding a constraint.
13. Sweep Without Touching: Manipulate specific objects without touching a constraint.
14. Same Texture: Recognize objects with the same texture and manipulate them into a specific container.
15. Same Shape: Recognize objects with the same shape and manipulate them into a specific container.
16. Manipulate the Old Neighbor: Place an object into a specific container, then place an object in the direction of its original location into the same container.
17. Pick in Order then Restore: Manipulate one object into different containers sequentially, and finally restore it to the initial container.

C.2. Real-world Tasks

In the real-world setting, a robotic arm with a gripper is utilized. A camera is positioned parallel to the robotic arm for optimal viewing. Real task experiments focus on object localization, object attribute recognition, complex scenes, complex reasoning, and contextual memory.

Additional tasks in the real-world setting include:

1. Visual Manipulation: It is the same as the VIMABench.
2. Same Shape: It is the same as the VIMABench.
3. Same Color: Recognize objects with the same color and manipulate them into a specific container.
4. Manipulate Old Neighbor: It is the same as the VIMABench.
5. Interfering Manipulation: There is a disturbance object in the environment and the robot need recognize the state. First distract the interfered object, then execute the following instruction.

D. Planning Template

We construct a template to prompt multimodal generator to generate planning. The complete prompt is shown in Figure 7.

E. Supplementary Result

The VIMABench contains 17 tasks of L1-L4 levels. We present the detailed experimental result in Table 8.

Method	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	Avg
L1 Level																		
Gato	79.0	68.0	91.5	57.0	44.5	54.0	74.0	-	18.0	-	61.0	88.5	-	-	83.5	33.5	2.5	58.1
Flamingo	56.0	58.5	63.0	48.5	38.0	48.5	62.5	-	3.5	-	66.5	86.0	-	-	40.0	43.5	2.5	47.5
GPT	62.0	57.5	41.0	55.5	45.5	47.5	54.5	-	8.5	-	77.0	81.5	-	-	41.0	38.0	0.5	46.9
VIMA	100.0	100.0	99.5	100.0	56.5	100.0	100.0	-	18.0	-	77.0	93.0	-	-	97.0	76.5	43.0	81.6
RT-2	100.0	98.0	97.0	58.0	30.0	98.0	97.0		14.0		84.0	79.0			93.0	52.0	47.0	72.8
CaP	90.0	80.8	96.0	65.3	61.3	80.0	84.7	-	38.0	-	68.0	60.0	-	-	67.3	58.0	76.0	71.2
VisualProg	92.0	27.0	9.0	29.0	90.0	38.0	87.0	-	21.3	-	65.3	30.7	-	-	92.0	36.7	28.7	49.7
I2A	91.3	81.4	98.2	78.5	72.0	82.0	88.0	-	42.0	-	72.0	68.0	-	-	78.0	64.0	85.2	77.0
RoboMP^2	100.0	89.3	100.0	92.7	95.3	86.0	96.0	-	55.3		89.3	71.3	-	-	86.0	98.7	96.7	**89.0**
L2 Level																		
Gato	56.5	53.5	88.0	55.5	43.5	55.5	53.0	-	14.0	-	63.0	90.5	-	-	81.5	33.0	4.0	53.2
Flamingo	51.0	52.5	61.5	49.5	38.5	47.5	55.5	-	5.5	-	70.5	82.0	-	-	42.0	39.0	3.0	46.0
GPT	52.0	52.0	49.5	54.5	45.5	52.5	51.0	-	11.0	-	76.5	84.0	-	-	43.0	38.0	0.5	46.9
VIMA	100.0	100.0	99.5	100.0	54.5	100.0	100.0	-	17.5	-	77.0	93.0	-	-	98.5	75.0	45.0	81.5
RT-2	100.0	96.0	97.0	56.0	27.0	95.0	97.0		10.0		84.0	83.0			92.0	43.0	34.0	70.3
CaP	90.0	79.3	96.0	64.7	60.0	74.6	85.3	-	37.3	-	66.7	62.7	-	-	66.0	52.7	74.7	70.0
VisualProg	84.0	26.0	11.3	38.7	87.3	30.0	80.7	-	20.0	-	71.3	22.0	-	-	94.7	24.0	30.0	47.7
I2A	91.5	80.8	97.8	74.9	69.5	81.0	86.0	-	44.0	-	70.5	65.0	-	-	80.0	66.0	84.0	76.2
RoboMP^2	99.3	78.7	100.0	91.3	91.3	82.0	92.0	-	36.0		92.0	68.7	-	-	91.3	96.7	96.7	**85.9**
L3 Level																		
Gato	51.0	58.0	84.5	56.5	35.5	53.5	49.0	-	15.0	-	65.0	-	-	-	52.0	33.0	0.0	43.5
Flamingo	49.0	50.0	66.5	47.0	35.0	47.5	50.0	-	4.0	-	66.0	-	-	-	30.5	43.5	0.5	40.8
GPT	52.0	51.0	55.0	49.5	40.0	46.0	50.5	-	5.0	-	82.0	-	-	-	37.0	38.0	1.5	42.3
VIMA	99.0	100.0	100.0	97.0	58.0	100.0	99.0	-	17.5	-	90.5	-	-	-	97.5	46.0	43.5	79.0
RT-2	96.0	94.0	96.0	52.0	31.0	95.0	93.0		11.0		97.0				93.0	40.0	3.0	66.8
CaP	90.0	79.3	95.3	63.3	60.0	74.0	84.7	-	37.3	-	66.0	-	-	-	64.3	51.3	72.0	69.8
VisualProg	83.3	25.3	13.3	27.3	62.0	32.0	80.7	-	17.3	-	70.0	-	-	-	73.3	24.0	39.7	45.7
I2A	91.8	80.2	97.4	81.8	65.8	79.0	89.0	-	38.0	-	71.0	-	-	-	78.0	62.0	82.0	76.3
RoboMP^2	100.0	83.3	100.0	86.0	86.0	78.7	95.3	-	40.7	-	92.0	-	-	-	89.3	98.0	92.7	**86.8**
L4 Level																		
Gato	-	-	-	-	-	-	-	20.5	-	0.0	-	-	0.0	29.0	-	-	-	12.4
Flamingo	-	-	-	-	-	-	-	21.0	-	0.0	-	-	0.0	27.5	-	-	-	12.1
GPT	-	-	-	-	-	-	-	20.5	-	0.5	-	-	0.0	36.0	-	-	-	14.3
VIMA	-	-	-	-	-	-	-	100.0	-	0.0	-	-	0.0	95.5	-	-	-	48.9
RT-2	-	-	-	-	-	-	-	98.0	-	0.0	-	-	0.0	90.0	-	-	-	47.0
CaP	-	-	-	-	-	-	-	74.0	-	28.0	-	-	0.0	76.7	-	-	-	44.7
VisualProg	-	-	-	-	-	-	-	52.0	-	64.0	-	-	0.0	92.7	-	-	-	52.2
I2A	-	-	-	-	-	-	-	84.0	-	35.0	-	-	0.0	80.0	-	-	-	49.8
RoboMP^2	-	-	-	-	-	-	-	76.7	-	98.7	-	-	0.0	96.7	-	-	-	**68.0**

Table 8: The full result of all tasks on VIMABench. “-” denotes that the task is excluded from this level.

```
THIRD PARTY TOOLS:
------
You have access to the following tools:

# Libraries
from PIL import Image
import numpy as np
import scipy
import torch
import cv2
import math
from typing import Union

IMPLEMENTED TOOLS:
------
You have access to the following tools:

# First Level: File IO
templates = {} # dictionary to store and cache the multi-modality instruction
# possible keys in templates: "scene", "prompt_images"
# NOTE: the word in one instruction inside {} stands for the visual part of the instruction and will be obtained with get
operation
# Example: {scene} -> templates.get('scene')
BOUNDS = {} # dictionary to store action space boundary

def GetObsImage(obs, view='top') -> Image.Image:
"""Get the current image to start the system. The default view is top view.
Examples:
image = GetObsImage(obs)
"""
pass

…

# Second Level: Core Modules
## Perception Modules
class GCMP:
def generate(image: np.ndarray, query: str) -> list:
"""Generate the coordinate of query objects according to the image"""
pass

def PickPlace(pick: np.ndarray, place: np.ndarray, bounds: np.ndarray, yaw_angle_degree: float = None, tool: str =
"suction") -> str:
"""Pick and place the object based on given locations and bounds"""
pass

…


Examples:
------
Use the following examples to understand tools:
## Example 1
# Instruction: Put the {object1} into the {object2}.
def main_1() -> dict:
"""Execute the given instructions of placing the object1 into the object2"""
image = GetObsImage(obs)
drag_obj_loc = GCMP.generate(image=image, query='object1')
base_obj_loc = GCMP.generate(image=image, query='object2')
action = PickPlace(pick=drag_obj_loc, place=base_obj_loc, bounds=BOUNDS)
info = RobotExecution(action=action)
return info

## Example 2
# Instruction: Put the {texture1} object in {scene} into the {texture2} object.
def main_2() -> dict:
"""Execute the given instructions of placing texture1 object in scene into the texture2 object"""
image = GetObsImage(obs)
image_scene = GetPromptImage(templates.get('scene'))
obs_obj_dict = GCMP.generate(image=image, query='all objects in the scene')
scene_obj_dict = GCMP.generate(image=image_scene, query='all objects in the scene')
drag_obj_loc = SelectFromScene(obs_obj_dict=obs_obj_dict, scene_obj_dict=scene_obj_dict, 'texture1')
base_obj_loc = GCMP.generate(image=image, query='the texture2 object')
action = PickPlace(pick=drag_obj_loc, place=base_obj_loc, bounds=BOUNDS)
info = RobotExecution(action=action)
return info


Begin to execute the task:
------
Please solve the following instruction step-by-step. You should ONLY implement the main() function and output in the
Python-code style. Except the code block, output fewer lines.

Instruction: dark yellow and blue polka dot squre is blicket than yellow and blue polka dot squre. Put the {adv} blicket
{object1} into the {object2}.
```

Figure 7: Full template