



**university of
groningen**

**faculty of science
and engineering**

Visually Grounded Large Language Models (VGLLMs) for Robotic Manipulation in Unknown Environments

Juan Luis López González

December 10, 2024



**university of
 groningen**

**faculty of science
 and engineering**

University of Groningen

**Visually grounded large language models (VGLLMs) for robotic manipulation
 in unknown environments**

MASTER'S THESIS

To fulfill the requirements for the degree of
 Master of Science in Artificial Intelligence
 at University of Groningen under the supervision of

**Georgios Tzifas,
 Prof. Tsegaye Misikir Tashu
 and
 Prof. Hamidreza Kasaei**

Juan Luis López González (s5156750)

December 10, 2024

Abstract

This thesis investigates the potential of Visually Grounded Large Language Models (VGLLMs) for few-shot robotic tabletop manipulation tasks. Six models (Kosmos-2, QWEN-VL, Florence-2, Gemini-1.5-Pro, Gemini-1.5-Flash, and GPT4-o) were evaluated on their zero-shot grounding capabilities on a subset OCID, HOTS and a novel PyBullet dataset from a simulation environment. Uniquely, this study examines the ability of five VGLLMs to generate grounded robotic API calls from diverse natural language instructions referencing objects and target locations, via few-shot prompting of interleaved images, instructions, and expected grounded API calls. A novel robotic pipeline is proposed that unifies the traditionally separate stages of visual grounding and code generation within a single VGLLM pass, enabling users to instruct a robot to perform pick-and-place actions using natural language. The findings show that while most models can generate visually grounded text consistently, there is room for improvement on their performance. Zero-shot grounding achieved a maximum Intersection over Union (IoU) score of approximately 50% on the datasets. Few-shot performance, while enabling grounded API call generation, achieved a maximum IoU of around 30%. In the simulated environment, the best performing model achieved a 30% success rate for grasping the correct object. Despite current limitations, this work highlights the potential of VGLLMs to unify robotic pipelines and motivates future research into fine-tuning these models with grounded robotic data.

Contents

List of Figures	III
List of Tables	IV
1 Introduction	1
1.1 Overview	1
1.2 Research Questions	1
1.3 Outline	1
2 Related Work	3
2.1 LLMs and Multimodal LLMs	3
2.2 Visually Grounding LLMs	4
2.3 Robotics with LLMs and VLMs	4
3 Methodology	6
3.1 Robotic Environment	6
3.2 Datasets	6
3.3 Models and Implementation	8
3.3.1 VGLLM Models	8
3.3.2 Other Models Used	10
3.3.3 Base and Proposed Environment	12
3.4 Prompt Engineering	13
3.5 Evaluation Metrics	14
4 Experiments and Results	17
4.1 Zero-Shot Grounding Performance	17
4.1.1 OCID Dataset	17
4.1.2 Pybullet Robotic Simulation Dataset	17
4.1.3 HOTS Dataset	20
4.1.4 Prompting experiments	21
4.1.5 Few-Shot Grounding for Robotic Planning on Pybullet	21
4.2 Unified Robotic Pipeline Performance	23
4.2.1 Grasp Success Rate	23
5 Discussion	28
5.1 Grounding Capabilities	28
5.2 Grounding and Planning with Few-Shot prompting	28
5.3 Robotic Simulation	29
5.4 Future Work	32
References	33
Appendices	36
A Figures	36

List of Figures

1	Ambiguity problem example.	4
2	Environment Configurations	6
3	Florence-2 model architecture, input-output examples for different tasks. Image taken from (Xiao et al., 2023).	10
4	GR-Convnet architecture taken from (Kumra, Joshi, & Sahin, 2021)	12
5	Segment Anything Model. Image adapted from (Kirillov et al., 2023).	12
6	Base robotic pipeline provided, using modular approach for object grounding.	13
7	Proposed robotic pipeline using VGLLMs for object grounding and robotic manipulation.	13
8	Average IoU scores for grounding per model.	18
9	Averaged IoU scores over bounding box and segmentation masks for each model per prompt type on the PyBullet Dataset.	24
10	Average success rates for grasping the correct object over all the query scenarios on the simulation experiment.	24
11	Rates of grasping the correct object, grasping an object, and the certainty of grasping the correct object over grasped ones on the simulation experiment.	27
12	GroundingDINO example use with Stable Diffusion.	36
13	PyBullet Dataset samples.	36
14	Examples of instances from the OCID dataset with the bounding box and segmentation mask of the object highlighted in red.	39
15	Grounding predictions samples for all models on the OCID dataset (Zoom in).	40
16	Random Grounding predictions samples for all models on the OCID dataset (Zoom in).	41
17	Grounding predictions samples for all models on the PyBullet dataset (Zoom in).	42
18	Random Grounding predictions samples for all models on the PyBullet dataset (Zoom in).	43
19	Grounding predictions samples for all models on the HOTS dataset (Zoom in).	44
20	Random Grounding predictions samples for all models on the HOTS dataset (Zoom in).	44
21	Grounding predictions for GPT4o, Gemini-1.5-flash and Gemini-1.5-pro-002 models on the Name Relative query type on the PyBullet Simulation (Zoom in).	45
22	Grounding predictions for KOSMOS-2 and QWEN-VL models on the Name Relative query type on the PyBullet Simulation (Zoom in).	46
23	Grounding predictions samples for all models per query type on the PyBullet Simulation (Zoom in).	47
24	Random Grounding predictions for all models per query type on the PyBullet Simulation (Zoom in).	48

List of Tables

1	Reference types from the OCID dataset.	7
2	List of YCB objects utilized in the robotic environment for robotic simulation and synthetic dataset creation.	7
3	List of query type categories used on the PyBullet simulation dataset.	7
4	Reference types from the HOTS dataset.	8
5	Overview of the VGLLM Models Selected for Evaluation	11
6	Prompts used for grounding sentences per model.	14
7	Examples of expected responses generated for the few-shot prompting.	14
8	Expected responses from query instructions using QWEN-VL for robotic planning.	16
9	Segmentation mask IoU for grounding a sentence from the OCID dataset.	19
10	Bounding box IoU for grounding a sentence from the OCID dataset.	19
11	Segmentation mask IoU for grounding robotic instructions to an object from the PyBullet dataset per model.	20
12	Bounding box IoU for grounding robotic instructions to an object from the PyBullet dataset per model.	21
13	Segmentation IoU for grounding sentences to an object from the HOTS dataset per model.	21
14	Bounding box IoU for grounding sentences to an object from the HOTS dataset per model.	22
15	Segmentation mask IoU using QWEN-VL model with different prompts on the OCID dataset.	22
16	Masks IoU on grounding different query instruction types on the PyBullet Dataset. When the query type includes targets, then the IoU is calculated for the relative object target position.	25
17	Bounding Box IoU on grounding different query instruction types on the PyBullet Dataset. When the query type includes targets, then the IoU is calculated for the relative object target position.	26
18	Success rates for grasping the correct object given a robotic instruction from different query types per model.	26
19	Prompting styles tested with KOSMOS-2 for few-shot grounding.	30
20	Exception counts of malformed robotic API calls on the simulation per model and per query type.	30
21	Samples of Generated API calls that produced an exception.	31
22	Number of target bounding boxes generated on relative query types per model on the PyBullet simulation experiments.	31
23	Number of target positions not mapped in the simulation per query type per model from the generated VGLLM responses.	32
24	Object attributes from the HOTS dataset	37

1 Introduction

1.1 Overview

The vision of autonomous and intelligent robots seamlessly integrated into everyday life, from handling household chores to transforming industrial processes, is rapidly approaching. Advancements in computing, algorithms, and large-scale datasets have democratized AI, making complex models accessible even via smartphones. These models fuel our optimism for the arrival of an Artificial General Intelligence (AGI), by their generalization skills across domains. However, the transition from theoretical breakthroughs to robust real-world robotic systems presents substantial challenges. These include the scarcity and lack of diversity in real-world robotic data, the critical need for safety in robot deployments, the limitations of real-time performance, and the inherent complexity of translating natural language instructions into robot actions. This thesis addresses some of these challenges by exploring the potential of foundational visually grounded large language models (VGLLMs) to be used in robotic pipelines that could unify visual grounding and planning. VGLLMs offer a unique paradigm: the ability to directly associate text with specific image regions. This approach offers a pathway to creating more efficient, robust, and adaptable robotic systems. These referring capabilities are being transferred to more use cases as it enhances the interaction with the models, for example as in conditional image generation from grounded text bounding boxes in (Y. Li et al., 2023). There is no surprise such referring capabilities will come in handy on the robotics setting.

1.2 Research Questions

This thesis investigates the potential of VGLLMs for robotic manipulation. By directly grounding language instructions within visual contexts, VGLLMs offer a pathway toward more unified robotic pipelines. This work looks to answer the following research questions:

1. How accurate are VGLLMs grounding image regions within expected robotic scenes using zero-shot prompting across different types of sentences?
2. How effectively are VGLLMs grounding image regions while generating robotic API calls from user instructions, using few-shot prompting with interleaved multimodal examples?
3. Can VGLLMs effectively replace traditional LLM + external visual grounding pipelines for a more unified and efficient robotic manipulation approach?

To address these questions, we evaluate the performance of six different models on three datasets: two real world dataset and a synthetic one. Datasets are used to assess the capabilities of grounding through zero-shot prompting. To evaluate the performance of grounding while planning with few-shot prompt strategies, only the synthetic dataset is used. Finally we propose a robotic pipeline, unifying both grounding and planning based on a natural language instruction to assess the potential of current VGLLMs in robotics.

1.3 Outline

The remainder of this thesis is organized as follows: Chapter 2 provides a comprehensive overview of relevant work on LLMs, VLMs, and their use in robotic grounding and planning. Chapter 3 details the methodology, including the datasets descriptions, model selection, implementation details of the

benchmarks and robotic pipeline, and the evaluation metrics. Chapter 4 presents the experimental results obtained from applying VGLLMs to robotic planning tasks. Chapter 5 discusses these results, considering their implications, limitations, and potential improvements and suggestions for future work.

2 Related Work

This chapter reviews some relevant foundational concepts and prior research relevant to this thesis. First, by discussing recent advancements on Large Language Models (LLMs) and their multimodal extensions, such as Visual Language Models (VLMs). Then, by discussing a special type of VLMs we call Visually Grounded LLMs (VGLLMs) and their unique capabilities of referencing text and image regions. Finally, this chapter provides context on current use cases of LLMs and VLMs in robotics, highlighting the potential of using VGLLMs to unify external grounding and text generation for planning within a single model.

2.1 LLMs and Multimodal LLMs

Large Language Models (LLMs) have revolutionized the field of Artificial Intelligence with their impressive capabilities in natural language processing. These novel models are built upon variants of the transformer architecture (Vaswani et al., 2023), which has allowed to handle long context tasks. LLMs are trained using massive web-scale datasets, allowing them to achieve remarkable performance on different tasks such as in summarization, translation, and question answering (QA), among others. The field continues to advance rapidly, with new models and benchmarks emerging constantly in the LM arena (Chiang et al., 2024). LLMs are foundation models because of their ability to generalize to new tasks not available in their training datasets through zero and few-shot prompting strategies. These abilities makes us wonder the potential for their use in a wide range of applications across a wide array of domains, including robotics (Kaddour et al., 2023).

Despite the impressive capabilities of LLMs, early models lack the ability to process data other than text. Failing to ingest different data modalities limit the capabilities of LLMs to solve problems in different domains. This limitation motivated the development of Multimodal LLMs (MMLLMs), which can handle diverse input modalities beyond text. LLMs with inputs in addition to text are called Multi-modal LLMs (MMLLMs). Visual Language Models (VLMs) are a type of MMLLMs trained on web-scale image-text datasets, integrating visual understanding and semantic knowledge together. Similar to LLMs, VLMs demonstrate strong generalization and foundation capabilities through zero-shot and few-shot prompting strategies. VLMs incorporate visual information into the same semantic embedding space with text, allowing to capture relationships between images and texts as seen in (H. Liu, Li, Wu, & Lee, 2023) and (Alayrac et al., 2022).

This integration allows to complete tasks that require information from both the visual and text inputs, such as in the Visual Question Answering benchmark and image captioning, in which models are tasked to answer questions about the images, and to generate descriptions about the image, respectively. A Visual Language Model (VLM) consist of a model with visual encoder and text encoder that are used to transform an input of both text and image modalities into a shared semantic embedding space. These embeddings are then normally used by a text decoder in an auto-regressive manner to predict the next token as usual. Sharing the embedding space, allows the completion of tasks such as visual question answering, image captioning, and classification. LLAVA from (H. Liu et al., 2023) was one of the first VLMs considering grounding text to image regions. LLAVA incorporated prompting techniques in one of their test sets, that provided visual context to questions via the description of bounding boxes of objects present on an image. VLMs trained on grounded image-text paired datasets (texts with references to an image region) are what this work refers to as Visually Grounding LLMs (VGLLMs).

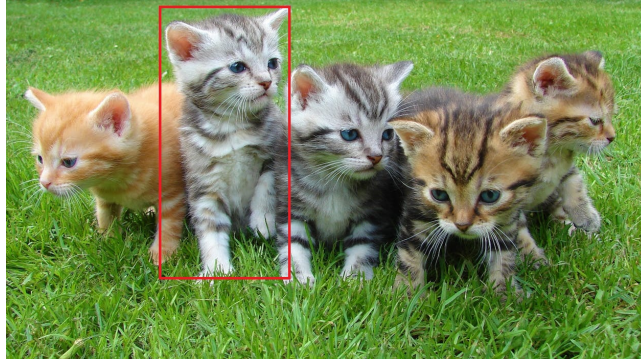


Figure 1: Ambiguity problem example.

2.2 Visually Grounding LLMs

Visually Grounded Large Language Models (VGLLMs) represent a significant advancement towards integrating visual grounding and language understanding within a single model. These models are trained on image-text pairs that are augmented with annotations of regions of interests. VGLLMs can both interpret and generate text that explicitly refers to specific image regions. In other words, they are able to take grounded text and images as input and provide grounded text as output. This is an important improvement to VLMs, since these type of models are capable of understanding explicit visual references from images to take as context for generating the next token. Additionally, VGLLMs are able to output the same explicit visual references and make a better interaction with the user or next prompting techniques. Visual grounding can be used to eliminate ambiguity, particularly when textual descriptions are insufficient or ambiguous. An example of this problem is shown in [Figure 1](#), where the text "The cutest cat" applied to an image with multiple cats is ambiguous, a bounding box clarifies the specific cat being referenced. Additionally, the VGLLM may respond with a reference to a region of the image as well, making the response unambiguous and more interactive. An example on how visual grounding can enhance interactions is from ([S. Liu et al., 2024](#)), where the authors use an open-set object detector that grounds expressions to object regions via bounding boxes. It integrates the object detector with Stable Diffusion ([Rombach, Blattmann, Lorenz, Esser, & Ommer, 2022](#)), by creating a pipeline with a selection prompt that maps an expression to an image region, then using a generation prompt to modify the image via the Stable Diffusion model, conditioned on the grounded bounding box as seen in [Figure 12](#). An example of a VGLLM is ([Peng et al., 2023](#)), one of the models studied. KOSMOS-2 was trained on grounded image-text pairs and is capable of generating text and referring objects to specific locations of an image through special localization tokens added to the vocabulary. ([Bai et al., 2023](#)) is another model used in our study, that follows a similar approach with a variability on their region vocabulary. Florence-2 ([Xiao et al., 2023](#)) another model used in this study, is capable of grounding to regions on the image but with the capability of not only grounding through bounding boxes, but also other types such as polygon representations.

2.3 Robotics with LLMs and VLMs

LLMs have been trained in all kinds of data sources, in a web-scaled manner. This has raised up the question to how the world knowledge learned by these models could be transferred to solve different tasks in various applications. As seen in ([Kaddour et al., 2023](#)), LLMs are used as chat-bots in various industries, computational biology, computer programming, creative work, law, medicine, reasoning, social sciences, synthetic data generation, and robotics. The capacity of LLMs to process and

understand complex sequences of information, coupled with their potential for generalization from the world knowledge present in web-scale training data has fueled research into their application in robotics, specifically in decision-making, planning and control. LLMs opened up the possibility of humans interacting with robots using natural language expressions, simplifying technical knowledge for control. In addition of the use cases of LLMS, VLMs add visual understanding which could further condition the decision-making, planning, and control of the robots using them.

LLMs and VLMs have been used in robotic task planning and control successfully. For example, (Huang, Abbeel, Pathak, & Mordatch, 2022) investigates if the world knowledge in the training data in LLMs is useful to act in interactive environments by grounding high level tasks to a set of actionable action steps. (Singh et al., 2022) with ProgPrompt, utilizes LLMs to decompose a high level task into sub-tasks by generating code style function calls, from a set of available actions or functions, use examples, and the set of objects in the environment. SayCan from (Ahn et al., 2022) utilizes a language model to decompose high-level tasks, by combining the probabilities of a skill being useful (according to the LLM) and the probabilities of a value function per skill (trained with the probability of successfully executing the skill) to select the skill to perform, grounding the instruction to the environment. Code as policies from (Liang et al., 2023) writes robot policies with code from a natural language instruction, having access to perception functions, control primitives, recursively generate code for undefined functions to ultimately generalize to new tasks. (Ha, Florence, & Song, 2023) utilizes an LLM to decompose tasks into subtasks, grounding the subtasks into robotic API code instructions, and create functions to check for success conditions. GraspGPT (Tang, Huang, Ge, Liu, & Zhang, 2023) uses LLMs to generate novel concepts from a natural language instruction, then uses this novel concept relating it to concepts from training data, and finally generalizes by grounding a task oriented grasping skill from the known concept to the novel one (A know concept could be to pour a cup and the novel could be to dispense a pitcher). PIVOT from (Nasiriany et al., 2024) utilizes a VLM to choose between a set of proposed robotic actions that are annotated visually to an image, similar to what SoM prompting from (Yang et al., 2023) does to help the VLM solve tasks by pointing out annotations added to the input image, but in a robotic environment. PaLM-E from (Driess et al., 2023) is a VLM, that has been injected with continuous, embodied observations like images, state estimates, and other sensor data into the language embedding space of a pre-trained language model. They did this by encoding continuous observations into a sequence of vectors with the same dimension as the embedding space from the base language model. Built on top of PALM-E, (Brohan et al., 2023) with RT-2, a vision language action model (VLA), that is like an VLM that directly outputs robotic actions. The model generates the action in words and an encoded positional and rotational changes, and gripper position. They fine-tuned a VLM with robotic trajectory data and vision-language tasks. AffordanceLLM from (Qian et al., 2024) leverages knowledge from LLMs, they take an image encoder and project its embeddings into the embedding space of a text encoder (a VLM), and produce a special $\langle \textit{mask_token} \rangle$, whose hidden state is then used as prompt to a decoder to generate a dense affordance map.

(J. Li et al., 2024) propose a new benchmark to evaluate multimodal LLMs in robotics (MMRo), totaling 14 different metrics in domains as perception, task planning, visual reasoning, and safety measurement. They found that is no model trustworthy enough to be used as core for all these benchmarks, each model had its own advantages and disadvantages. This one benchmark already started to question the effectiveness of visual grounding (part of the perception benchmarks), in robotics. This thesis contributes to this evolving field by investigating the potential use of VGLLMs to unify grounding and planning.

3 Methodology

This chapter describes the methodology used in this thesis. It describes the simulated robotic environment, the datasets used for evaluation, the selected VLLMs and other supporting models, the implementation of the the proposed robotic pipeline, the prompt engineering strategies, and the evaluation metrics. The source code use can be found in this GitHub repository. ¹

3.1 Robotic Environment

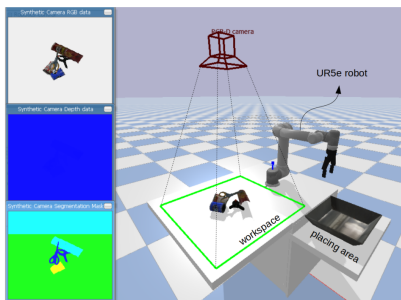
The robotic environment used for the experiments is a simulated tabletop setup, of a Universal Robot arm (UR5c) equipped with a two-fingered Robotiq 2F-140 gripper and an RGB-D camera in an eye-to-hand configuration mounted on top of the table. The simulation is implemented in PyBullet, adapted from previous work from (Oude Vrielink & Kasaei, 2021). The environment configuration is shown in Figure 2a. The robot arm is mounted on a side of a table, where objects are randomly placed. These objects are drawn from a subset of the YCB object dataset (Calli et al., 2015), and are listed on Table 2. A tray, located next to the robot arm, is used as a target location for placing objects. The base configuration of this setup was used to accept natural language instructions, which are then translated into robotic API calls to execute the desired manipulation tasks. Additionally, another environment is shown in Figure 2b, of a real robot to help with the perspective of the simulated environment.

3.2 Datasets

Three datasets were used to evaluate the performance of the selected VLLMS:

Object Clutter Indoor Dataset (OCID) (Suchi, Patten, Fischinger, & Vincze, 2019): This dataset is a subset of the original, consisting of 173 real-world images of cluttered scenes. Each image is paired with a sentence referencing an object within the scene. Objects are referenced in different categories: name, attribute, affordance, spatial relations, visual relations, semantic relations, and multi-hop relations. Table 1 lists examples to these reference types, the number of samples of each, and an example sentence. Additionally, each scene includes information about the segmentation mask and bounding box of the object of interest. Examples of scenes from the OCID dataset are shown in Figure 14.

¹Source code: <https://github.com/juanluislopez24/VLLMbased.robotic.manipulation>.



(a) Pybullet environment configuration.



(b) Real robot environment configuration.

Figure 2: Environment Configurations

Table 1: Reference types from the OCID dataset.

Reference Type	Number of Samples	Example
Semantic relations	13	Something to wipe myself with
Visual relations	19	Marker that has the same color as the stapler
Multi hop	24	First cereal from the left
Spatial relations	33	Bottle next to the binder
Affordance	16	I want to cook some spaghetti
Name	42	Feh package
Attribute	26	Towel with letters on it

PyBullet Simulation Dataset: This dataset was generated using the PyBullet robotic simulation environment. Objects used in the robotic environment are a subset from the YCB dataset (Calli et al., 2015). The objects comprised of 16 objects commonly found at home. The list of objects is shown in Table 2. Scenes contain 2 to 6 randomly selected objects. Ground truth segmentation masks, obtained from the simulator, were used to derive accurate bounding boxes for each object. A natural language instruction generator was used to create diverse instructions referencing objects and target locations, using the corresponding reference types and target types, which used in combination, this thesis refers to query types. The resulting query types are shown in Table 3. Each generated instruction includes information about the target object, location, and other relevant details for evaluation. Examples of scenes from the PyBullet generated dataset are shown in Figure 13.

Table 2: List of YCB objects utilized in the robotic environment for robotic simulation and synthetic dataset creation.

Scissors	Gelatin Box	Cracker Box	Strawberry
Power Drill	Chips Can	Foam Brick	Potted Meat Can
Tennis Ball	Medium Clamp	Banana	Pear
Master Chef Can	Tomato Soup Can	Mustard Bottle	Hammer

Table 3: List of query type categories used on the PyBullet simulation dataset.

Name Tray	Attribute Tray	Affordance Tray
Name Table	Attribute Table	Category Tray
Name Relative	Attribute Relative	

Household Objects in Tabletop Scenarios (HOTS): This dataset comprises of 118 tabletop scenes featuring 46 different objects with various attributes associated to them. Scenes include an RGB image with its corresponding object labels, segmentation masks and bounding boxes. This dataset was used previously in (Tziafas & Kasaei, 2023) to enhance the interpretability and interactivity in robot manipulation. This dataset was collected using the real-world robotic environment shown in Figure 2b. The object attributes include their category, super-category, color, and material. The

objects attributes are shown in Table 24. To align with the OCID dataset format, a sentence generator was developed to produce natural language references for objects within the HOTS scenes, using the provided object attributes and spatial relationships derived from the bounding boxes. These generated sentences include some of the same reference types as the OCID dataset: name, spatial relations, visual relations, attribute, and affordance. Details of the reference types present in this dataset are shown in Table 4.

Table 4: Reference types from the HOTS dataset.

Reference Type	Number of Samples	Example
Semantic relations	118	A red fruit to keep the doctor away.
Visual relations	118	The pen the same color as the hot pringles
Spatial relations	118	The plastic item on the bottom of the cassis can
Affordance	118	I need something to eat my cereal from
Name	118	Fanta can
Attribute	118	Something black made of metal for eating.

3.3 Models and Implementation

This section covers details around the models used and details the implementations around the dataset evaluation and the proposed robotic pipeline.

3.3.1 VLLM Models

Six models were selected for evaluation (Table 5). These models were chosen for their potential to combine visual grounding with text.

KOSMOS-2 (Peng et al., 2023): is a multi modal large language model, that is capable of perceiving object descriptions, such as bounding boxes, and grounding text to an image. Authors say that their work is a foundation for the development of Embodiment AI, converging language, multimodal perception, action, and world modeling. KOSMOS-2 is a transformer-based causal language model, built upon KOSMOS 1, which incorporates grounding and referring capabilities. They introduce a new dataset comprised of grounded image-text pairs (GRIT). Bounding boxes are discretized into location tokens, and then appended to the text spans using a "hyperlink" format. The output, when the model's answers includes a reference to an image region, utilizes the same location tokens. The models format to encoding text and image regions is shown in Listing 1. They say that grounding capability enables to get more accurate, informative, and comprehensive responses. The model architecture includes a vision encoder with 24 layers with 1024 hidden size and 4096 FFN (Feed forward network) intermediate size. The multimodal LLM is a 24 layer MAgneto Transformer (H. Wang et al., 2022), 32 attention heads, 8192 FFN intermediate size. The total trainable parameter size is around 1.6B. Image input dimension is 224×224 with patch size of 14×14 . The image is divided into 32 bins consisting of 7×7 pixels. This means that 32×32 location tokens where added to the vocabulary. Kosmos-2 uses Komos-1 weights for initialization, with the new vocabulary randomly initialized. All parameters were updated during training and instruction tuning.

Qwen-VL (Bai et al., 2023): is a vision-language model combining a large language model (LLM) with a vision encoder and a position-aware adapter that processes both text and images. The architecture consists of a Qwen-7B 7.7B language model, a 1.9B ViT based image encoder, and 0.08B a

Listing 1: Kosmos-2 grounding example verbatim

```
This is an image of a <phrase>tennis ball </phrase>
<object><patch_index_0120><patch_index_0201 ></object>
next to a pair of <phrase>scissors </phrase>
<object><patch_index_0378><patch_index_0476 ></object>
on top of a table .
```

Listing 2: QWEN-VL grounding example verbatim

```
The <ref>potted meat can </ref><box >(776,245),(928,361) </box>
is placed next to a
<ref>bonfire </ref><box >(202,303),(495,632) </box>
```

vision-language adapter that compresses the visual features and adds positional information that helps for visual understanding. The complete architecture totals 9.6B parameters. The model first was pre-trained on a general image-text dataset, then trained on a multi-task annotated dataset, and finally instruction fine-tuned for user interaction. During training the model is capable to learn grounding by taking in information from region descriptions, questions, and bounding boxes. They normalized and encoded bounding box coordinates as text strings, and combined with the other text data it is capable of associating words and sentences with its corresponding regions on the image. An example of this is shown in [Listing 2](#). This model showed great potential for zero-shot image captioning and VQA benchmarks.

Florence-2 ([Xiao et al., 2023](#)): is a multitask model trained with extensive visual annotations. The model includes an image encoder and a multi-modality encoder-decoder. They introduce a dataset called FLD-5B with 5.4B annotations on 126M different images, which was created autonomously. Florence-2 is capable of object detection, captioning, and grounding. They adopt a sequence-to-sequence framework, treating tasks as a translation problem. Given an image and a task prompt, they generate the output. Depending on the task, the output is either text or region on the image. Grounding to regions is possible in three different ways: bounding box, quad box, and polygon representations. This is shown in [Figure 3](#). As the previous models, they extended the vocabulary to include location tokens. The vision encoder is from the DaViT architecture ([Ding et al., 2022](#)). Images are transformed into visual token embeddings. The multi-modality encoder decoder uses a standard encoder-decoder transformer architecture that processes both visual and language token embeddings. The model's architecture is shown in [Figure 3](#).

Gemini 1.5 Pro and Gemini 1.5 Flash ([Team et al., 2024](#)): are set of multimodal models from Google capable of handling text, visual, and audio inputs. These models are natively multimodal, so they can support input mix from different modalities. These models are built to handle long context windows of up to 10 million tokens. Gemini 1.5 Pro is a sparse mixture-of-expert (MoE) Transformer-based model. A MoE model has a routing capability to use the inputs on a subset of the model's parameters when performing inference. The Flash variant is a transformer decoder model with multimodal capabilities as well.

GPT4o ([OpenAI et al., 2024](#)): is a multimodal model from OpenAI, capable of also handling text, visual, and audio inputs. It is also capable of supporting a mix of the different modalities. GPT4o 128k context window still lacks behind Google's. There exists a smaller version of the model called GPT4o-mini, but is not used in this project. GPT-4 model is a mixture-of-experts of around 1.76

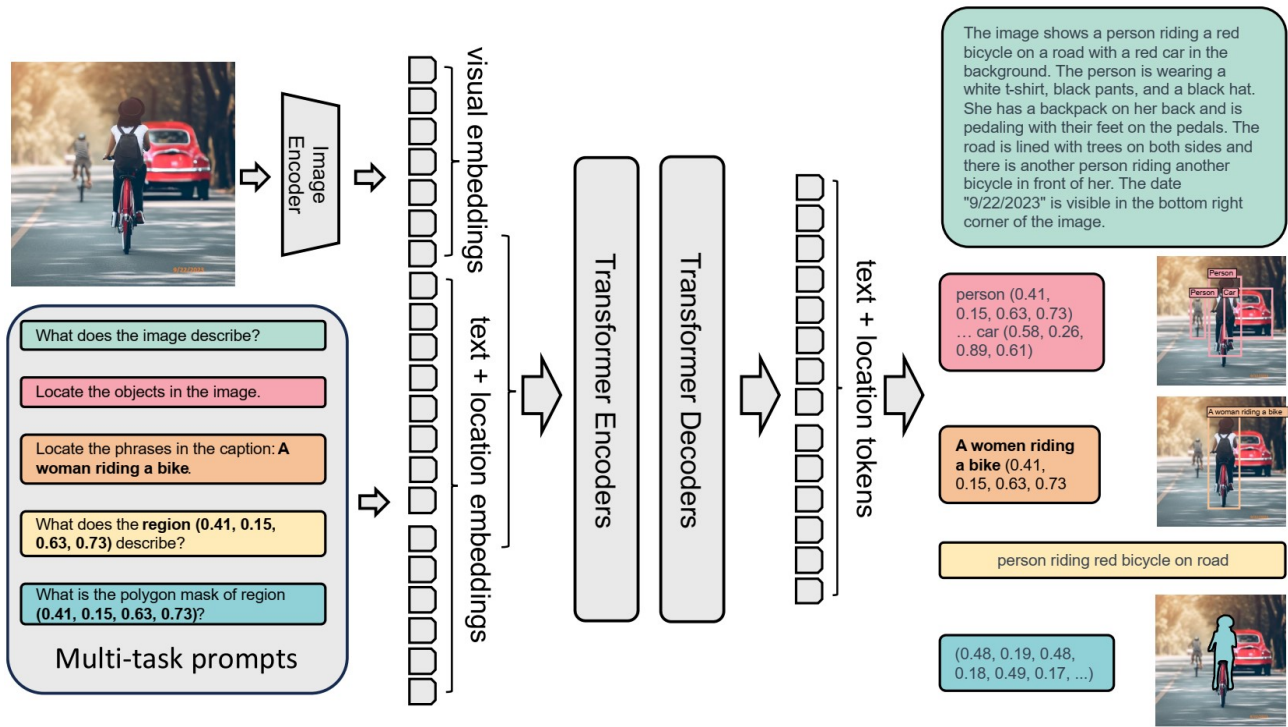


Figure 3: Florence-2 model architecture, input-output examples for different tasks. Image taken from (Xiao et al., 2023).

trillion parameters, but there is no public information about the parameter size of GPT4o, although it is also suspected to be a MoE. Additionally, GPT4-o is capable of generating images and audio, but it is unclear if it is part of the same model architecture, or if they are using connectors behind the scenes.

3.3.2 Other Models Used

GR-Convnet is a Generative Residual Convolutional Neural Network model that generates antipodal grasps (Kumra et al., 2021). The network takes as input RGB and Depth images to generate pixel-wise grasps in the form of three different generated images. The input is processed and then passed through three convolutional layers, followed by five residual layers, later three transposed convolutional layers that generate 4 different images. These 4 images correspond to a pixel-wise grasp quality score, angles in the form of $\cos 2\theta$ and $\sin 2\theta$, and the width of the end effector. The angle images are then combined to form the required angle for the grasp pose. The GR-Convnet architecture is shown in Figure 4. This network is used in the proposed robotic pipelines, shown in Figure 6 and Figure 7, to generate the best grasping poses for the objects to be manipulated. In the pipeline, there is a function called *grasp_object_from_mask*, which takes as parameters the mask and the RGB depth image from the top of the table and returns grasp pose estimation results. The grasp results are filtered using the mask and the best grasping poses are chosen according to their quality (outputted by GR-Convnet).

Segment Anything Model (SAM) is a foundation model for segmentation (Kirillov et al., 2023). It is a promptable model trained on a large dataset, enabling generalization. The model has flexibility on how it can be prompted, including point, box, mask, and initial results on text prompts. The model's architecture consist of an image encoder, a prompt encoder, and a mask decoder as shown in

Table 5: Overview of the VGLLM Models Selected for Evaluation

Model	Architecture Overview	Grounding Mechanism	Training Data
Kosmos-2	Vision Encoder, Multi-modal LLM (1.6B)	Bounding Box through added vocabulary for location tokens	GRIT: grounded image-text pairs, monomodal text corpora, image-caption pairs, and interleaved image-text data.
Qwen-VL	Vision encoder, vision-language adapter, LLM (9.6B)	Bounding box tokens encoded as strings and added vocabulary	Large-scale web-crawled Chinese and English image-text pairs, GRIT, and other grounding datasets
Florence-2	Vision Encoder and a multi-modality encoder-decoder (0.77B)	Bounding box, quad box, polygon from strings and added vocabulary	FLD-5B: Image-text pairs, region-text pairs, text-phrase-region triplets.
Gemini Pro	Multi-modal transformer-based Mixture-of-Experts (MoE)	Via prompting (Bounding box JSON or Tuple in our case)	Google’s proprietary multi-modal dataset
Gemini 1.5 Flash	Multi-modal transformer-based encoder-decoder	Via prompting (Bounding box JSON or Tuple in our case)	Google’s proprietary multi-modal dataset
GPT-4o	Multi-modal transformer-based encoder-decoder	Via prompting (Bounding box JSON or Tuple in our case)	OpenAI’s proprietary multi-modal dataset

Figure 5. The output of SAM can include multiple valid masks associated with their corresponding confidence scores, in the environment and dataset evaluations, we consider only the mask with the highest score. The image encoder utilizes MAE (Masked Auto-encoders) (He et al., 2021) with ViT (Vision Transformer) (Dosovitskiy et al., 2021). The resulting image encoding can be re-utilized on different prompts. The prompt encoder deals with two types of prompts: sparse and dense. Sparse prompts include points, boxes and text. Dense include the mask prompts. Points and boxes are represented by positional encodings summed with learned embeddings for each prompt type. Text prompts are encoded using CLIP’s encoder (Radford et al., 2021). Mask prompts are encoded using convolutions and then summed element-wise to the image embedding. The mask decoder maps the image embedding, the prompts, and an output token to a mask. The decoder uses bi-directional cross-attention to update all embeddings. After two blocks, the image is up-sampled and a MLP is used to calculate the probability of a position being part of the foreground. The model outputs 3 masks, additionally the model predicts a confidence score for each.

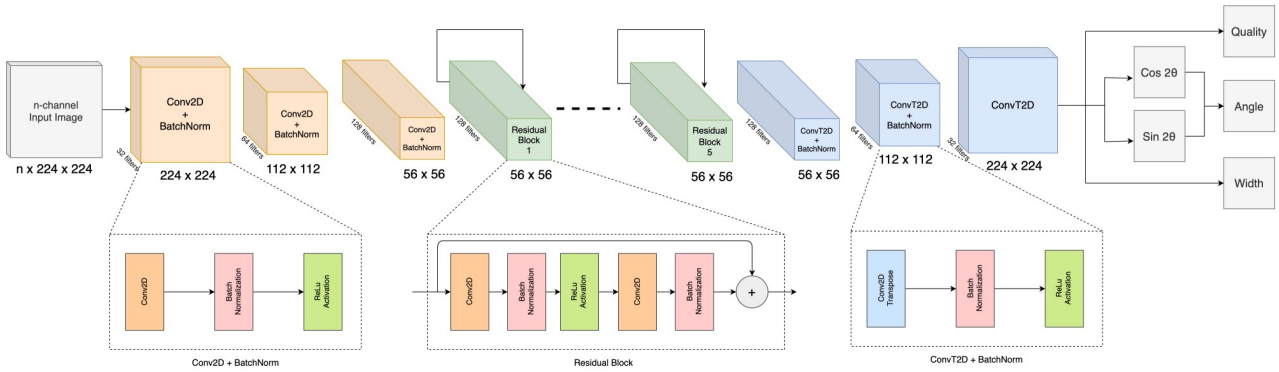


Figure 4: GR-Convnet architecture taken from (Kumra et al., 2021)

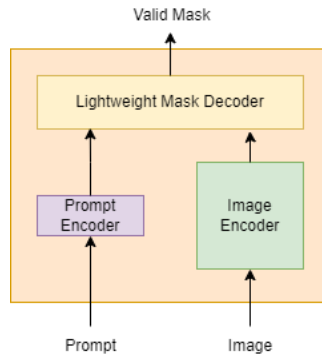


Figure 5: Segment Anything Model. Image adapted from (Kirillov et al., 2023).

3.3.3 Base and Proposed Environment

The base environment was provided by the supervisors, being modified from (Oude Vrielink & Kasaei, 2021). The environment was modified so that it was capable of receiving open text instructions from a user, generate robotic API instructions and execute them. As soon as the environment is instantiated with all the corresponding objects and elements, the following happens: First, the ground truth segmentation is used to classify each object according to a list of categories, all used by CLIP (Radford et al., 2021). CLIP takes an image of the object and matches it to the category which resembles the most from the list. Then for each object classified, a number grasp poses are calculated using GR-Convnet (Kumra et al., 2021). After having mapped categories and possible grasps to objects in the environment, the interface asks the user for a natural language instruction. The natural instruction is processed by a BLOOM (Workshop et al., 2023), an LLM that is used to generate robotic API calls from the user's input. These API calls are then executed accordingly. One important thing to mention is that the labels used to classify the objects with CLIP, are already grounded to grasp poses, which are what BLOOM passes as parameters when generating the robotic API calls. The robotic API includes trajectory planning and execution routines. The overview of the base pipeline is shown in Figure 6.

The proposed pipeline leverages VGLLMs to unify the external object grounding from CLIP and the generation of robotic API calls from BLOOM in a single pass of the model. The pipeline works in the following steps: First, when the objects are instantiated, the user is prompted for an instruction. Then, the image and the instruction are interleaved on a prompt and passed to the corresponding VGLLM as input. The VGLLM outputs a sequence of *move* API calls, alongside the respective object references as bounding boxes (In the same API call as part of the parameters). The object references are used

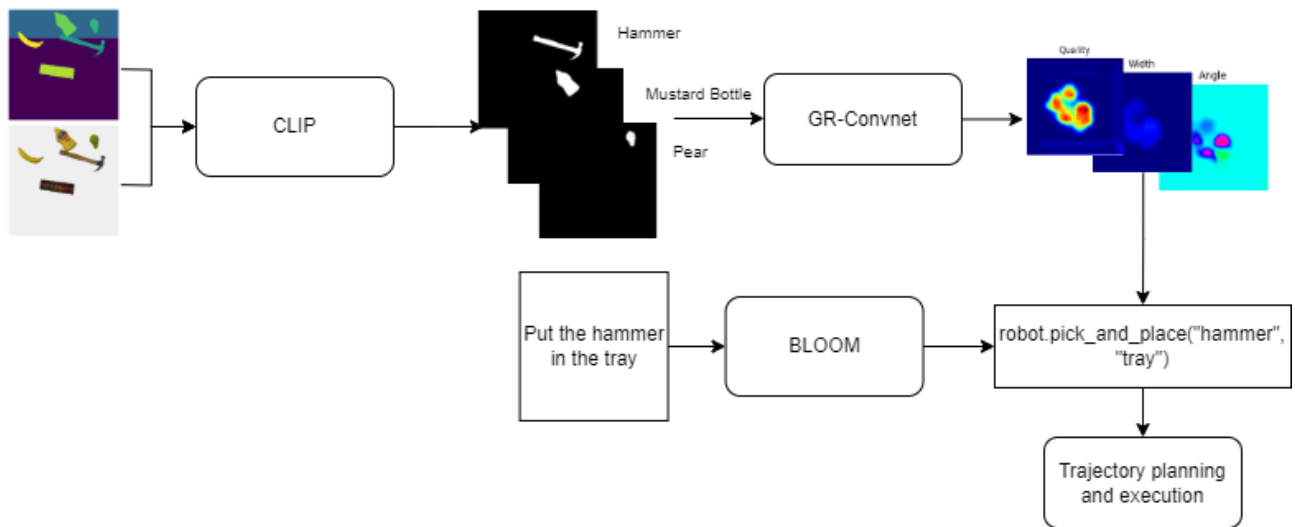


Figure 6: Base robotic pipeline provided, using modular approach for object grounding.

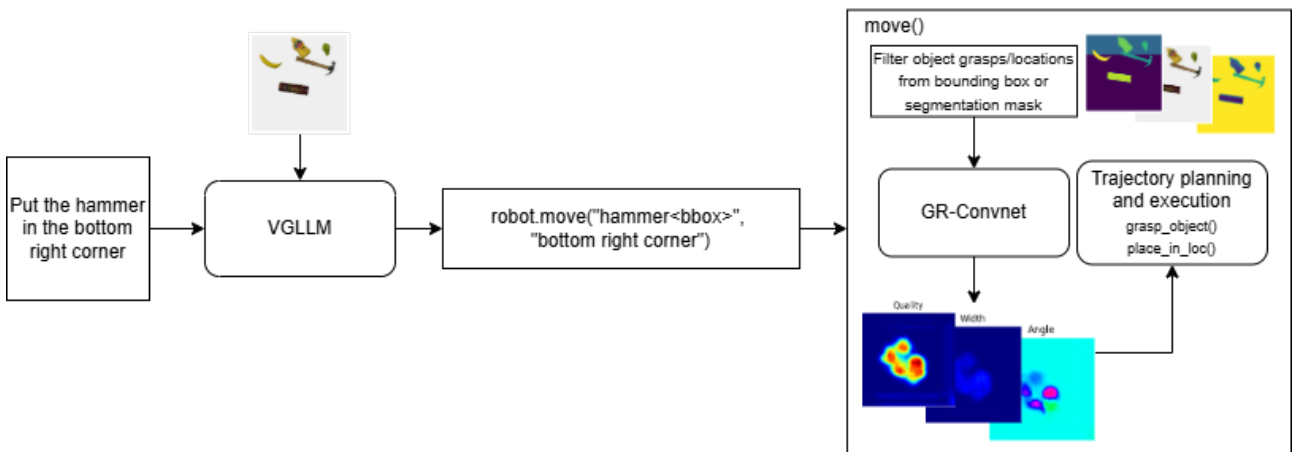


Figure 7: Proposed robotic pipeline using VGLLMs for object grounding and robotic manipulation.

to get the appropriate mask by utilizing the box prompting method with SAM (Kirillov et al., 2023). The obtained mask is then used to calculate the possible grasping poses. Finally, robotic API calls for trajectory planning and execution routines are used. The overview of the proposed robotic pipeline is shown in Figure 7.

3.4 Prompt Engineering

Model-specific prompt engineering was crucial for obtaining proper inference results on both grounding, grounding with robotic API calls, and generating the robotic API calls on the simulation. Manual testing through trial and error of multiple prompting strategies was used per model. The prompts for grounding focused on guiding the models to ground the entire input sentence. The zero-shot prompt templates per model for grounding on all three different datasets are listed on Table 6. These templates replace the word *sentence* with the appropriate text referencing an object or instruction.

For grounding while generating robotic API calls, a few-shot prompting strategy was utilized. Using the PyBullet dataset, a function was used to create an expected generated response for each of the models, using each model's unique way to ground image regions. The PyBullet dataset provides

Table 6: Prompts used for grounding sentences per model.

Model	Prompt
QWEN-VL	<ref> <i>sentence</i> </ref>
KOSMOS-2	<phrase> <i>sentence</i> </phrase>
Florence-2	<OPEN_VOCABULARY_DETECTION> <i>sentence</i>
Gemini 1.5 pro, Gemini 1.5 Flash, GPT4o	Provide the bounding box coordinates of the object in the image the following expression refers to: <i>sentence</i> Coordinates must be in the range between 0 and 1000. Do not round up the coordinates. Answer using this JSON schema: { "x_min": int, "y_min": int, "x_max": int, "y_max": int }

Table 7: Examples of expected responses generated for the few-shot prompting.

Model	Expected Response
GPT4o	<code>robot.move((45, 21, 91, 100), "left side")</code>
Gemini Models	<code>robot.move((816, 321, 968, 424), "tray")</code>
KOSMOS-2	<code>robot.move("tomato soup can", "tray")</code>
QWEN-VL	<code>robot.move("<ref>tomato soup can</ref><box>(789,318), (971,434)</box>", "center")</code>

the necessary ground truth information to do this for each type of instruction. An example of the generated expected responses for each model can be seen in [Table 7](#). These expected responses were used in the few-shot prompting strategy, that allowed the models to generate consistent robotic API calls. The few-shot prompt consists of interleaved images, natural language robotic instructions, and expected grounded robotic API calls. Additionally, some examples of the expected QWEN-VL model responses are shown in [Table 8](#) for each query type. In order to make the models robust to the different query type instructions, a mixed prompt was used in both the grounding evaluations and on the robotic simulation. The mixed prompt consisted on two samples of each query type, totaling 14 examples. Additionally, a specialized prompt was used for the grounding evaluations, comprised of 14 examples of the same query type. An example of a mixed interleaved prompt for the Gemini models is shown in [Listing 3](#).

3.5 Evaluation Metrics

The metric for evaluating grounding performance, both zero-shot and few-shot, is the Intersection over Union (IoU) between predicted and ground truth bounding boxes and segmentation masks (Equation 1). One important thing to mention is that the segmentation masks are generated with SAM (Kirillov et al., 2023) using the generated bounding boxes from the VGLLMs. IoU is a metric that

allows to quantify the overlap between the ground truth regions of an image and the predicted segmentation masks or bounding boxes. For the PyBullet dataset's few-shot experiments, IoU was calculated for both the object the instruction refers to and the relative target object (when applicable). For robotic simulation experiments, the success rate of grasping the correct object was measured. Additionally, a precision metric, defined as the number of correctly grasping the desired object divided by the total number of times an object was grasped. This measure is used to assess the certainty of correctly grasping an object when an object is grasped.

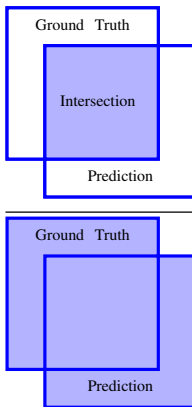
$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{Intersection}}{\text{Ground Truth} \cup \text{Prediction}} \quad (1)$$


Table 8: Expected responses from query instructions using QWEN-VL for robotic planning.

Query and Query Type	Qwen-VL Expected Response
Query: Carry the potted meat can to the tray Type: name tray	robot.move("<ref>potted meat can</ref> <box>(776,245), (928,361)</box>", "tray")
Query: Carry the gelatin box to the upper left corner Type: name table	robot.move("<ref>gelatin box</ref><box>(254,758), (379,821)</box>", "upper left corner")
Query: Put the tennis ball to the right of the scissors Type: name relative	robot.move("<ref>tennis ball</ref><box>(812,366), (897,450)</box>", "<ref>scissors</ref><box>(258,285), (495,486)</box>", "right")
Query: Move the fruit that is often eaten raw to the tray Type: attribute tray	robot.move("<ref>pear</ref><box>(281,491), (366,616)</box>", "tray")
Query: Relocate the container for potato chips to the left side Type: attribute table	robot.move("<ref>chips can</ref><box>(522,0), (736,312)</box>", "left side")
Query: Relocate the packaging for a food item to the right of the banana Type: attribute relative	robot.move("<ref>chips can</ref><box>(566,312), (888,437)</box>", "<ref>banana</ref><box>(611,781), (857,875)</box>", "right")
Query: I need to make a quick and easy meal Type: affordance tray	robot.move("<ref>potted meat can</ref><box>(566,727), (660,875)</box>", "tray")
Query: Carry all the food in the tray Type: category tray	robot.move("<ref>pear</ref><box>(250,303), (370,388)</box>", "tray") robot.move("<ref>chips can</ref><box>(651,406), (982,526)</box>", "tray")

4 Experiments and Results

This section presents the results of the experiments evaluating the performance of the chosen models: KOSMOS-2, QWEN-VL, FLORENCE-2, Gemini-1.5-pro-002, Gemini-1.5-flash, and GPT4-o. Grounding capabilities of the models were measured on three datasets: The Object Clutter Indoor Dataset (OCID) and the robotic PyBullet simulation dataset, and the Household Objects in Tabletop Scenarios (HOTS). Both zero-shot grounding capabilities (OCID, PyBullet, HOTS) and few-shot grounding during robotic API Call generation (PyBullet) are assessed. The evaluation metrics for the experiments are Intersection over Union (IoU) for bounding boxes and segmentation masks (Using SAM with the bounding box as prompt input). Finally, we analyze the performance of our proposed unified robotic pipeline, utilizing the few-shot prompting generating grounded robotic instructions measuring success rates for grasping.

4.1 Zero-Shot Grounding Performance

These experiments evaluate the models' ability to ground objects in images without any explicit examples or training in the provided prompts.

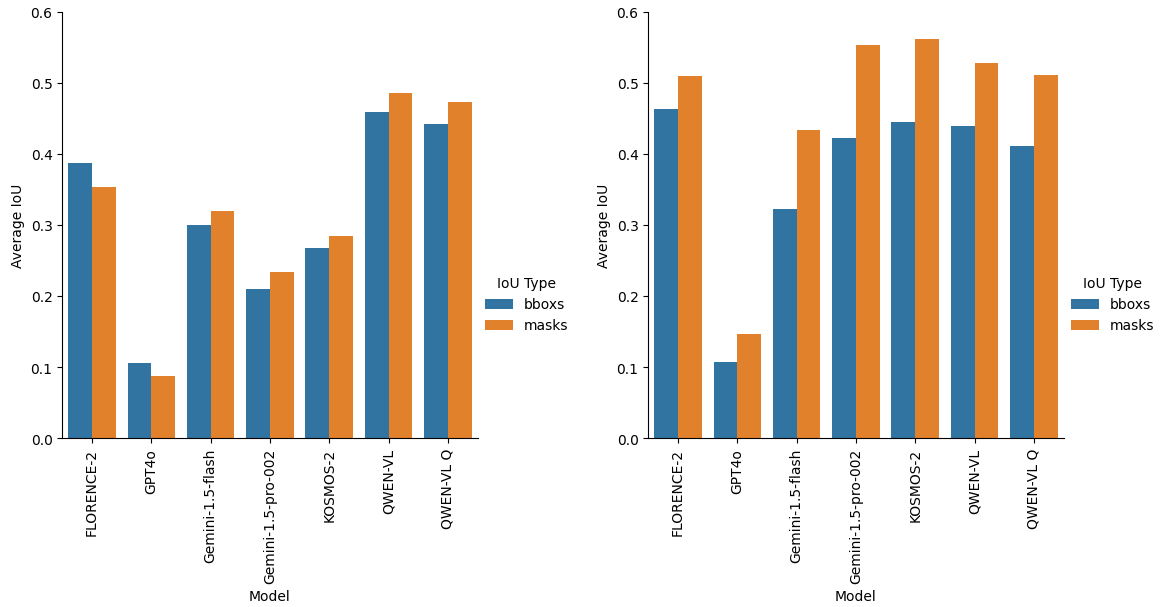
4.1.1 OCID Dataset

The zero-shot grounding results on the OCID dataset are shown in Tables 9 and 10 for segmentation masks and bounding boxes, respectively. The results are broken down by the type of object reference used in the sentence (Affordance, Attribute, Multi-hop, Name, Semantic Relations, Spatial Relations, and Visual Relations). This breakdown reveals how model performance varies with different ways of referring to objects. The results show that the IoU scores for the bounding box and segmentation masks produced by SAM (Kirillov et al., 2023), appear to be consistent to one another. The exact same models share the highest scores on both IoU measures across the different referral types of sentences. QWEN-VL models are consistent on their grounding performance, being the best performing models on six out of the seven different types of sentences on the OCID dataset with an average score of 47% and 45% (QWEN-VL and QWEN-VL Q). Followed by FLORENCE-2 with 37%, Gemini-1.5-flash with 31%, KOSMOS-2 with 27%, Gemini-1.5-pro with 22%, and finally GPT4-o with 9% scores. GPT4o was behind the other models in all the cases. The average scores per model on the bounding box and segmentation mask IoUs can be seen in Figure 8a. Qualitative results are shown in Figure 15 and Figure 16.

For the Affordance type of sentence, the quantized version of QWEN-VL excelled, with 40% IoU, compared to the 26% achieved by the full precision model. The Attribute type of sentence received one of the highest IoU score measures with FLORENCE-2 achieving an average IoU score close to 69%, followed by the two QWEN-VL and QWEN-VL quantized with an IoU score of 61% and 57%, respectively. On the multi-hop, name, semantic relations, and visual relations types of sentences, QWEN-VL variants outperformed the rest. The performance of FLORENCE-2 on the name type of sentences was very close, with an average IoU score of 61%.

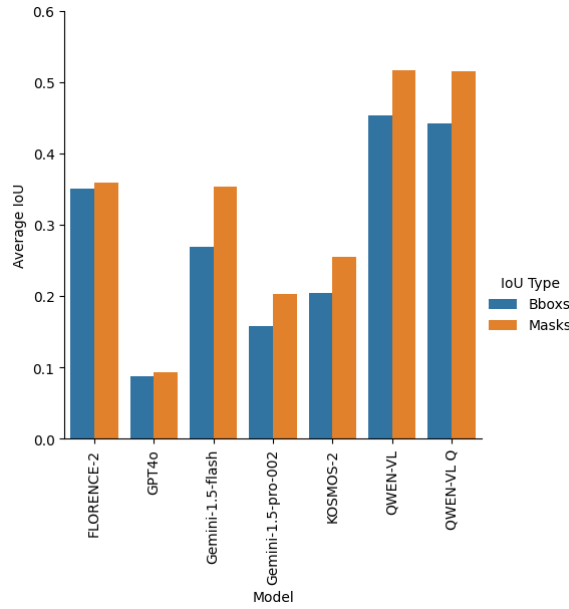
4.1.2 Pybullet Robotic Simulation Dataset

Table 11 and Table 12 show the IoU results for segmentation masks and bounding boxes, respectively, for each query type on the synthetic dataset. Here we are measuring the performance of grounding based on different query types or expressions that could be given to a robot. The query types include a combination of a way to refer the object to (similar to what was done with the OCID dataset) and



(a) OCID dataset

(b) PyBullet dataset



(c) HOTS dataset

Figure 8: Average IoU scores for grounding per model.

Table 9: Segmentation mask IoU for grounding a sentence from the OCID dataset.

	Affordance	Attribute	Multi Hop	Name	Semantic Relations	Spatial Relations	Visual Relations
FLORENCE-2	0.111639	0.671126	0.196845	0.593188	0.271794	0.265697	0.364090
GPT4o	0.140137	0.031564	0.151037	0.077607	0.077607	0.095448	0.044111
Gemini-1.5-flash	0.203507	0.396500	0.267592	0.437188	0.336308	0.301147	0.294352
Gemini-1.5-pro-002	0.318785	0.264089	0.157720	0.327945	0.162620	0.157604	0.247936
KOSMOS-2	0.312743	0.316129	0.128383	0.401764	0.274944	0.285859	0.269241
QWEN-VL	0.269836	0.626821	0.422539	0.667331	0.454686	0.532095	0.427783
QWEN-VL Q	0.406320	0.589831	0.372122	0.629653	0.387211	0.535048	0.389365

Table 10: Bounding box IoU for grounding a sentence from the OCID dataset.

	Affordance	Attribute	Multi Hop	Name	Semantic Relations	Spatial Relations	Visual Relations
FLORENCE-2	0.148479	0.707124	0.230261	0.634691	0.298831	0.305047	0.382490
GPT4o	0.139036	0.081969	0.094159	0.121245	0.119203	0.117818	0.066584
Gemini-1.5-flash	0.198703	0.369197	0.247035	0.374125	0.308639	0.305087	0.302775
Gemini-1.5-pro-002	0.276387	0.238015	0.145813	0.285311	0.156388	0.150366	0.219212
KOSMOS-2	0.287478	0.311535	0.123244	0.375885	0.259591	0.264009	0.248137
QWEN-VL	0.181505	0.609935	0.405175	0.645662	0.428542	0.512704	0.430601
QWEN-VL Q	0.328141	0.556802	0.357474	0.595979	0.326211	0.524050	0.410353

Table 11: Segmentation mask IoU for grounding robotic instructions to an object from the PyBullet dataset per model.

	Affor- dance Tray	Attribute Relative	Attribute Table	Attribute Tray	Name Relative	Name Table	Name Tray
FLORENCE-2	0.195985	0.343201	0.566884	0.550471	0.516010	0.723365	0.670068
GPT4o	0.169561	0.135780	0.145501	0.150403	0.108205	0.124894	0.192031
Gemini-1.5-flash	0.403804	0.367072	0.388034	0.472061	0.475603	0.465725	0.461623
Gemini-1.5-pro-002	0.425946	0.532825	0.528381	0.586545	0.547549	0.594905	0.654741
KOSMOS-2	0.339714	0.574507	0.553300	0.583257	0.685791	0.628356	0.564683
QWEN-VL	0.172149	0.492496	0.528467	0.613879	0.622361	0.623481	0.638810
QWEN-VL Q	0.158056	0.457138	0.513950	0.557816	0.628087	0.620496	0.637425

the target location type. The query types are listed in [Table 3](#). The performance of the models to ground objects under different query types will allow us to identify further strengths and biases of the models capability of grounding under different types of robotic natural instructions. The results appear to be consistent to what happen with grounding the OCID dataset, the models having the highest scores on the bounding box IoU also have the highest scores on the segmentation masks IoU. This time, the model performing the best on average was KOSMOS-2 with a score of 50% overall, achieving the highest score on two out of the seven different query types ([Table 3](#)), having relative target positions. FLORENCE-2 achieved the highest scores on three of the different query types, Attribute Table, Name Table, and Name Tray. Despite QWEN-VL dominating the OCID dataset, on this experiment it only achieved the highest score on the Attribute Tray query type. Finally, Gemini variants performed the best for the Affordance Tray query type, with the pro variant achieving the best score. The average scores per model on the bounding box and segmentation mask IoUs can be seen in [Figure 8b](#). On the figure mentioned, one can appreciate that FLORENCE-2, QWEN-VL variants, KOSMOS-2 and Gemini-1.5-pro had similar performances overall, and that again GPT4-o achieved the worst one. One important variation from the previous section is that in this case, the results show a better IoU score for segmentation masks, rather than on the bounding box, averaging over all query types per model. Qualitative results are shown in [Figure 17](#) and [Figure 18](#).

4.1.3 HOTS Dataset

The the IoU results for the zero-shot grounding on the HOTS datasets are shown in [Table 13](#) and [Table 14](#). These correspond to the IoU scores of the segmentation masks and bounding box, respectively. We notice that QWEN-VL showed the best performance again, with an average IoU score of around 0.44. The results follow the same pattern shown from the OCID dataset in [subsection 4.1.1](#). These resemblance on the results may be because of the similarities of both the OCID dataset and the HOTS datasets, having real world images and tabletop settings. The average scores can be seen on [Figure 8c](#). In this case, IoU scores from the segmentation masks are higher for all the models, meaning that given the predicted bounding boxes, the object segmentation performed with SAM helped achieve a higher object coverage. QWEN-VL models showed better performance, achieving the maximum IoU score across the different reference types, except on the name reference type which max score was achieved by the Florence-2 model. QWEN-VL model achieved an average score of 48.5%, while on the other

Table 12: Bounding box IoU for grounding robotic instructions to an object from the PyBullet dataset per model.

	Affor- dance Tray	Attribute Relative	Attribute Table	Attribute Tray	Name Relative	Name Table	Name Tray
FLORENCE-2	0.190912	0.311566	0.508603	0.493671	0.480548	0.651691	0.605650
GPT4o	0.111722	0.100661	0.103202	0.106398	0.086323	0.099485	0.145302
Gemini-1.5-flash	0.290995	0.283847	0.291155	0.347395	0.365129	0.325416	0.356305
Gemini-1.5-pro-002	0.320934	0.399274	0.421304	0.453038	0.389245	0.457071	0.512282
KOSMOS-2	0.225148	0.463717	0.452809	0.473122	0.556634	0.496989	0.450913
QWEN-VL	0.146964	0.414956	0.444724	0.527675	0.526600	0.504144	0.513156
QWEN-VL Q	0.129350	0.385298	0.426121	0.455782	0.512520	0.467531	0.500564

Table 13: Segmentation IoU for grounding sentences to an object from the HOTS dataset per model.

Model	Affordance	Attribute	Name	Semantic Relations	Spatial Relations	Visual Relations
FLORENCE-2	0.202726	0.518143	0.569380	0.255050	0.165573	0.444319
GPT4o	0.107603	0.079613	0.116497	0.068004	0.068912	0.119341
Gemini-1.5-flash	0.445052	0.408787	0.311481	0.387878	0.293700	0.277125
Gemini-1.5-pro-002	0.254626	0.253692	0.216772	0.192504	0.174172	0.129592
KOSMOS-2	0.189882	0.299865	0.281282	0.174694	0.209664	0.373482
QWEN-VL	0.505766	0.628947	0.503549	0.504378	0.451169	0.506606
QWEN-VL Q	0.537401	0.638695	0.508424	0.487208	0.403080	0.518964

end GPT4-o achieved 9%. Gemini-flash performed better than the pro version, with an average score of 31% compared to 18%. This again follows a similar pattern found on the OCID dataset. Qualitative results are shown in [Figure 20](#) and [Figure 20](#).

4.1.4 Prompting experiments

We wondered how much the prompt could affect a model’s ability to ground. And in order to perform the experiments with the best grounding prompt, we conducted a series of experiments with QWEN-VL. [Table 15](#) presents a detailed breakdown of QWEN-VL’s performance under different prompt variations on the Full precision and the quantized models, on the different OCID sentence types. The quantized version shows no apparent overall decline in performance, but in the implementation it showed an increase in inference time of 2.5 times compared to the full precision model.

4.1.5 Few-Shot Grounding for Robotic Planning on Pybullet

This experiment is a variation of the previous one, incorporating few-shot prompting with examples of interleaved robotic instructions, images, and the expected robotic API code result. In this experiment

Table 14: Bounding box IoU for grounding sentences to an object from the HOTS dataset per model.

Model	Affordance	Attribute	Name	Semantic Relations	Spatial Relations	Visual Relations
FLORENCE-2	0.216006	0.498736	0.536432	0.261656	0.165891	0.426838
GPT4o	0.107313	0.089148	0.102691	0.061439	0.065309	0.103083
Gemini-1.5-flash	0.337305	0.316628	0.218036	0.291993	0.242072	0.208769
Gemini-1.5-pro-002	0.213327	0.191517	0.168856	0.159646	0.127636	0.088991
KOSMOS-2	0.165988	0.231368	0.212000	0.146389	0.178258	0.296600
QWEN-VL	0.446867	0.549830	0.410138	0.452726	0.409425	0.451344
QWEN-VL Q	0.462671	0.551000	0.405387	0.431576	0.355660	0.449872

Table 15: Segmentation mask IoU using QWEN-VL model with different prompts on the OCID dataset.

Prompt	Precision	Semantic relations	Visual relations	Multi hop
<i>sentence</i>	Full	0.328437	0.307952	0.418007
	Int-4	0.399084	0.264625	0.376359
Ground the following expression: <i>sentence</i>	Full	0.333003	0.375285	0.426658
	Int-4	0.330987	0.427970	0.389476
<ref> <i>sentence</i> </ref>	Full	0.450087	0.329063	0.423292
	Int-4	0.439668	0.377151	0.386263
Grounding of <ref> <i>sentence</i> </ref>	Full	0.436341	0.427960	0.408340
	Int-4	0.456386	0.378951	0.432370

Prompt	Precision	Spatial relations	Affordance	Name	Attribute
<i>sentence</i>	Full	0.510406	0.000000	0.663005	0.516673
	Int-4	0.588936	0.000000	0.592813	0.613375
Ground the following expression: <i>sentence</i>	Full	0.548465	0.000000	0.606584	0.584493
	Int-4	0.557988	0.110653	0.631831	0.571872
<ref> <i>sentence</i> </ref>	Full	0.555681	0.069773	0.635003	0.587053
	Int-4	0.589378	0.147213	0.593287	0.520467
Grounding of <ref> <i>sentence</i> </ref>	Full	0.525373	0.270752	0.643970	0.646636
	Int-4	0.549216	0.402852	0.618426	0.603021

only the grounding capabilities are measured while performing the robotic planning from robotic natural language instructions. [Table 16](#) and [Table 17](#) provide the IoU results for the segmentation masks and bounding boxes, respectively, on the synthetic dataset with few-shot prompting producing robotic API calls. The performance on the different query types is analyzed. The query types are the same as the ones from [section 4.1.2](#) and are listed on [Table 3](#). Another aspect studied here is the inclusion of "Mixed" and "Specialized" prompt types, which allows a comparison between the mode's adaptability from using mixed examples from the different query types in the the prompt or using the same query type examples in the prompt for each query type experiment. It is important to note that the consistency of the generated API calls is not measured yet, but will be discussed in the next [section 4.2](#). Finally, we include an additional measure in query types having a relative target, in which we expect two objects being grounded per API call. The overall performance per model for each prompt type, averaged over the segmentation masks and bounding box IoU scores over all query types is shown in [Figure 9](#). To our surprise, comparing to the previous experiments, the grounding performance of all the models when using few-shot prompting to produce a robotic planning through API calls code generation is degraded. QWEN-VL and KOSMOS-2 show the best performance average IoU scores of around 30% and 27%, respectively. The other three models, suffered the most degradation, compared to the grounding results from the PyBullet without few-shot examples. Gemini-1.5-flash, Gemini-1.5-pro, and GPT4o had an average IoU score of 7%, 13%, and 15%, respectively. Although these models had the worst scores, they were the most consistent across all the query types and target groundings.

Target IoU scores exhibit different interesting points. The first, KOSMOS-2 has an IoU score of 0% for both relative target cases, it appears to be unable to ground a second object from the instruction. The second one, is that all the proprietary models show consistency across their scores, and outperform the QWEN-VL model on the mixed scenarios. QWEN-VL outperforms when using the specialized prompts. Overall, mixed or specialized prompts show a varied behavior on the IoU scores as seen on [Figure 9](#). Specialized prompting only increases the IoU scores in a few target types for some models, and sometimes it even decreases the performance.

4.2 Unified Robotic Pipeline Performance

4.2.1 Grasp Success Rate

[Figure 10](#), [Figure 11](#) and [Table 18](#) present the success rates on the proposed unified robotic pipeline. [Table 18](#) shows the success rates for grasping the correct object given a robotic instruction from different query types per model, on different object instances (120 instances per query type). The figures provide a more visual representation of the overall success rates. In [Figure 10](#), QWEN-VL shows the best performance averaged over all the query types with a success rate of 30%, followed by Gemini-1.5-pro with a rate of 24%, GPT4o with a rate of 23%, Gemini-1.5-flash with a rate of 15% and lastly KOSMOS-2 with rate of 13%. QWEN-VL achieved not only the highest average rate, but also the highest rate on all the different query types as seen on [Table 18](#). KOSMOS-2 showed the worst scores on every query type, except on the Attribute Relative one, being underperformed only by Gemini-1.5-flash. If we were to extrapolate the results from [section 4.1.5](#), one would have expected KOSMOS-2 to perform better. The results from the other models follow some kind of correlation, QWEN-VL performing the best, GPT4-o and Gemini-1.5-pro show similar scores on both experiments and Gemini-1.5-flash continued having the worst results among them. Gemini-1.5-pro shows similar performance to QWEN-VL in query types that reference the object to be picked by attributes, but this needs to be further studied.

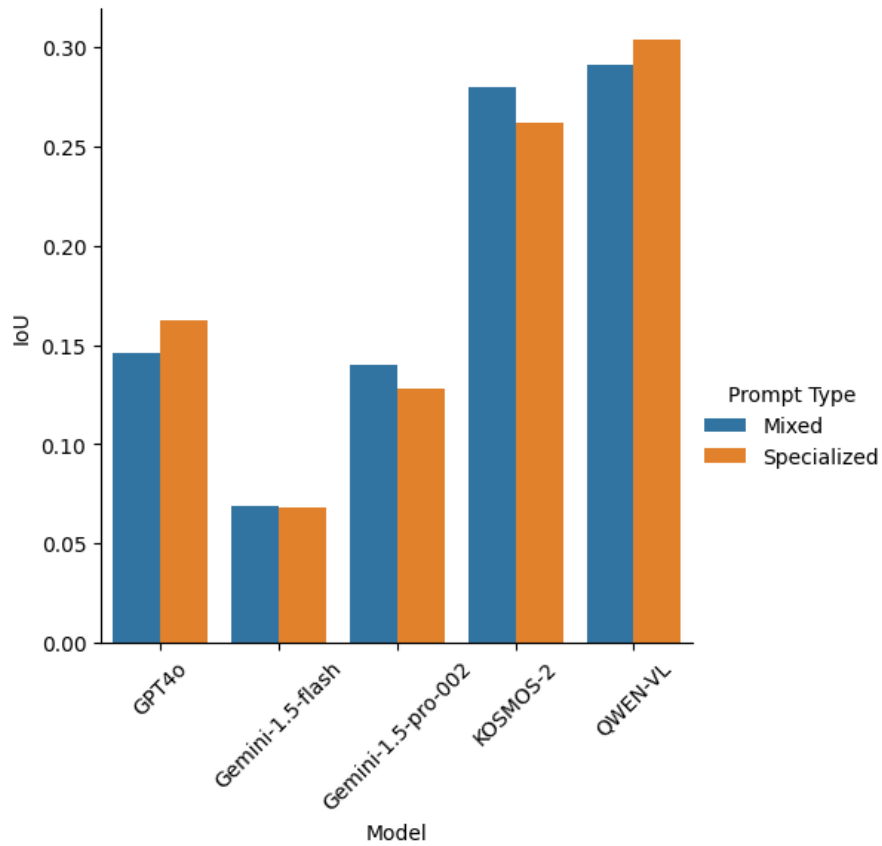


Figure 9: Averaged IoU scores over bounding box and segmentation masks for each model per prompt type on the PyBullet Dataset.

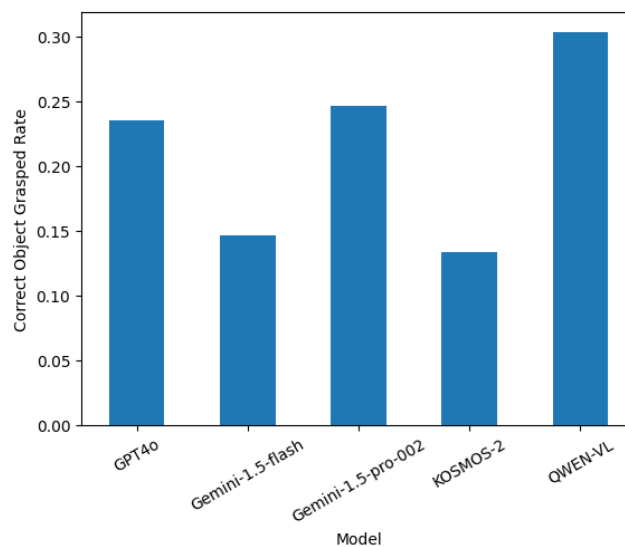


Figure 10: Average success rates for grasping the correct object over all the query scenarios on the simulation experiment.

Table 16: Masks IoU on grounding different query instruction types on the PyBullet Dataset. When the query type includes targets, then the IoU is calculated for the relative object target position.

Query Type	Prompt Type	Gemini-1.5 flash	Gemini-1.5 pro-002	GPT4o	KOSMOS-2	QWEN-VL
Affordance Tray	Mixed	0.051623	0.172131	0.144398	<u>0.287585</u>	0.236973
	Specialized	0.060114	0.144180	0.109295	0.219196	0.276851
Attribute Relative	Mixed	0.108381	0.192551	0.156625	<u>0.403313</u>	0.375721
	Specialized	0.120337	0.141535	0.216968	0.226795	0.268869
Target	Mixed	0.070102	0.130562	0.164428	0.000000	0.107091
	Specialized	0.109708	0.141310	<u>0.194501</u>	0.000000	0.152761
Attribute Table	Mixed	0.089497	0.129176	0.179090	0.370393	0.387000
	Specialized	0.098590	0.128313	0.173282	<u>0.396966</u>	0.347491
Attribute Tray	Mixed	0.091190	0.127346	0.168999	0.490268	0.399132
	Specialized	0.057545	0.144665	0.243847	<u>0.500812</u>	0.418654
Name Relative	Mixed	0.066159	0.178118	0.155518	0.388500	0.401556
	Specialized	0.061654	0.161117	0.209690	0.289828	<u>0.412664</u>
Target	Mixed	0.102100	0.182349	0.180869	0.000000	0.104501
	Specialized	0.097552	0.123354	0.177841	0.000000	<u>0.290771</u>
Name Table	Mixed	0.063733	0.204889	0.200875	0.428640	<u>0.495665</u>
	Specialized	0.073529	0.122353	0.195286	0.478497	0.416093
Name Tray	Mixed	0.095077	0.197602	0.244169	0.428712	0.384092
	Specialized	0.078963	0.207768	0.254386	<u>0.531280</u>	0.413270
Average		0.083103	0.157184	0.187226	0.302266	0.327175

Table 17: Bounding Box IoU on grounding different query instruction types on the PyBullet Dataset. When the query type includes targets, then the IoU is calculated for the relative object target position.

Query Type	Prompt Type	Gemini-1.5 flash	Gemini-1.5 pro-002	GPT4o	KOSMOS-2	QWEN-VL
Affordance Tray	Mixed	0.045986	0.109782	0.084938	0.217479	0.188066
	Specialized	0.049565	0.110600	0.080651	0.154092	0.233201
Attribute Relative	Mixed	0.071300	0.122913	0.103460	0.320188	0.313321
	Specialized	0.065269	0.098453	0.136705	0.165105	0.224148
Target	Mixed	0.051682	0.098714	0.098385	0.000000	0.085160
	Specialized	0.076652	0.118776	0.116871	0.000000	0.124862
Attribute Table	Mixed	0.067658	0.111528	0.122728	0.304277	0.316645
	Specialized	0.085120	0.102139	0.116258	0.324745	0.300577
Attribute Tray	Mixed	0.056105	0.098016	0.123175	0.395314	0.334281
	Specialized	0.055210	0.114128	0.146927	0.413812	0.342545
Name Relative	Mixed	0.040964	0.108050	0.125952	0.314303	0.333708
	Specialized	0.046987	0.108853	0.140785	0.215114	0.335854
Target	Mixed	0.061833	0.109062	0.099236	0.000000	0.071866
	Specialized	0.065759	0.095968	0.117791	0.000000	0.235146
Name Table	Mixed	0.042507	0.123239	0.123330	0.340432	0.395987
	Specialized	0.056645	0.103292	0.131340	0.377525	0.348985
Name Tray	Mixed	0.071809	0.123995	0.147854	0.343578	0.314808
	Specialized	0.067316	0.141754	0.162157	0.425601	0.328353
Average		0.059909	0.111070	0.121030	0.239531	0.268195

Table 18: Success rates for grasping the correct object given a robotic instruction from different query types per model.

Ref Type	Target Type	GPT4o	Gemini-1.5 flash	Gemini-1.5 pro-002	KOSMOS-2	QWEN-VL
Affordance	Tray	0.166667	0.075000	0.191667	0.150000	0.266667
Attribute	Relative	0.258333	0.133333	0.225000	0.175000	0.266667
	Table	0.208333	0.166667	0.283333	0.175000	0.300000
	Tray	0.250000	0.158333	0.266667	0.091667	0.300000
Name	Relative	0.241667	0.183333	0.250000	0.116667	0.275000
	Table	0.233333	0.158333	0.250000	0.141667	0.383333
	Tray	0.291667	0.150000	0.258333	0.083333	0.333333
Average		0.235714	0.146429	0.246429	0.133333	0.303571

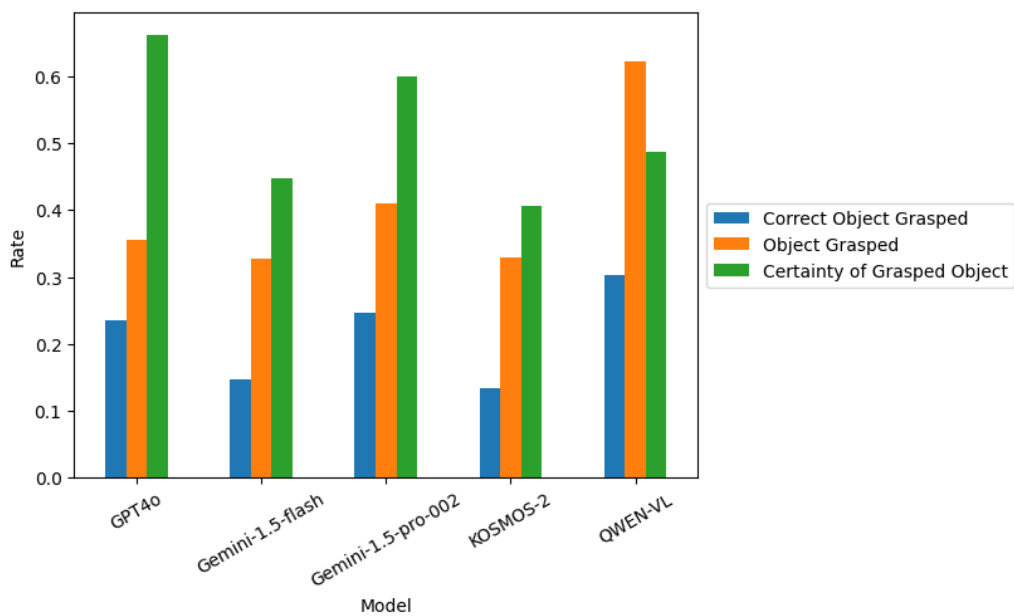


Figure 11: Rates of grasping the correct object, grasping an object, and the certainty of grasping the correct object over grasped ones on the simulation experiment.

5 Discussion

The experimental results demonstrate both the promise and limitations of VGLLMs for robotic planning. While these models exhibit a capacity for visual grounding, their performance in generating accurate and consistent robotic API calls from natural language instructions requires further investigation. In this section, results are discussed first by analyzing aspects about the grounding, in both zero and few-shot scenarios. Later, the aspects from the robotic simulation with the proposed pipeline.

5.1 Grounding Capabilities

Our zero-shot grounding experiments revealed that while all models demonstrated some ability to ground text to image regions, their performance varied significantly. Florence-2 achieved the best precision scores, particularly for sentences with object attributes and names on the OCID dataset and on names on the HOTS dataset. We believe that multi-task training for grounding could be beneficial for the model’s performance. As you recall, Florence-2 is able to ground text through bounding boxes, polygons, and quad-boxes as discussed in 3.3.1. The QWEN-VL models outperformed other models on the OCID dataset, but did fairly similar to KOSMOS-2, FLORENCE-2, and Gemini-1.5-pro models on the PyBullet dataset. The performance in grounding between the OCID and HOTS datasets is very similar, we think this is due to the similarities of the scenes and images.

The sample size of the OCID dataset used is too small to draw statistically significant conclusions about the model’s performance. In order to achieve grounding results from GPT4o and Gemini models, a zero-shot prompt was produced specifying the structure of the expected output in json format as seen in Table 6. Further study of the best prompt for grounding still requires further investigation, for example when using a naively formed prompt to simply ground through bounding box upper left corner and bottom right corner, GPT4o was using ranges between 0 and 1, but Gemini was using integers. We believe that these models were trained with some grounded text-image pairs, and that they are more capable of grounding with a specific format and value ranges. We can see that in (H. Liu et al., 2023), the VLM was prompted to understand grounded input in a specific format, testing for different grounding formats from training data and prompting from other studies could be fruitful.

5.2 Grounding and Planning with Few-Shot prompting

The generation of grounded robotic API calls via few-shot prompting was a success, but some challenges were revealed. For starters, the grounding IoU scores for the best performing models dropped from an average of around 50% on the grounding experiment, to 29% on the grounding while planning experiment. Another limitation we encountered was that FLORENCE-2, despite its strong zero-shot grounding performance from previous experiments, failed to output robotic API calls with few-shot examples through our best efforts on different prompts and was not included on this experiment. This is due to the fact that FLORENCE-2 is a small model and cannot be considered with the generalization capabilities of an LLM. KOSMOS-2 and QWEN-VL fell short on the grounding for relative object target positions. KOSMOS-2 was not able to generate a single API call with a bounding box on the target parameter on the robot.move function out of the 240 instances with relative object target positions. This was likely due to its difficulty on generating consistent robotic API instructions because it is also a model with small parameter sized decoder. Different prompts with intertwining the images, query instructions, and expected grounding text were tested, but despite our efforts, we were left using examples with no grounding tokens as seen in Table 7. QWEN-VL failed to generate a bounding box for target positions in only 8 out of the 240 instances. Grounding results

samples for the "Name Relative" query type are shown in [Figure 22](#). Generated texts never include an encoded bounding box on the second parameter of the move function. In contrast GPT4o and Gemini-1.5-flash correctly generated a target bounding box for all the 240 instances with query type with relative target positions, while Gemini-1.5-pro did for 237 instances. Grounding results samples for the "Name Relative" query type for these three models are shown in [Figure 21](#), where all samples include a blue bounding box, representing the target bounding box generated. Number of target bounding boxes generated per model are shown in [Table 22](#). Further investigation is warranted to find a few-shot prompting strategy for the KOSMOS-2 and QWEN-VL models to properly generate consistent API calls with relative target positions. The performance for the GPT4-o model improved significantly with few-shot examples, suggesting that these models can benefit from prior knowledge provided through prompting. The proprietary Gemini models displayed higher consistency across all tasks, particularly for relative object target grounding, demonstrating their potential for handling complex instructions. QWEN-VL and KOSMOS-2 achieved better performance than other models despite having significantly fewer parameters. This highlights the potential of efficient architectures, specifically trained on grounded image-text pairs, even in the context of few-shot learning. Mixed vs Specialized prompting did not show an overall difference on the performance, having a specialized prompt even decreased the model's grounding in some scenarios, meaning that the mixed prompt is more convenient for a more generalizable robotic behavior and grounding.

5.3 Robotic Simulation

QWEN-VL consistently achieved the highest success rates for grasping objects correctly, underscoring its better grounding and planning capabilities. GPT4-o and Gemini-Pro models exhibited around 6% lower success rates to QWEN-VL, but show that these models have significant potential. Both Gemini-Flash and KOSMOS-2 struggled in the simulation, emphasizing the need for further improvement in their grounding and API call generation accuracy. One of the reasons KOSMOS-2 is having such a bad grasping success rate is because it is failing to generate consistent robotic API calls. Despite of our efforts into different prompting strategies, we could not achieve consistent generation of correct API calls. Prompt options tested for KOSMOS-2 are described in [Table 19](#). KOSMOS-2 failed to output proper API calls on 49% of the instances, if we were to fix this issue we could have achieved a performance of around 26% of correctly grasping objects. QWEN-VL was another model that failed producing robotic API calls, but only at 4.6% of the times. Details about the counts on exceptions to robotic instructions generated per model and query type are shown in [Table 20](#). Some examples on the generated text that caused exceptions are shown in [Table 21](#). Examples in the table show that most exceptions occurred because the function call was not properly closed. On the QWEN-VL example, the model specially struggles with the Name Relative query type, as seen in [Table 20](#). GPT4-o and Gemini models did not suffer from generating a malformed robotic API call, showcasing consistency their text generation capabilities from larger models. Our certainty measure, Counts of correctly grasping an object divided by the counts of grasping an object, show that GPT4-o and Gemini models have a highest scores. As seen in [Figure 11](#), the green bar is the highest for GPT4-o with a certainty of 0.66 and Gemini-pro with 0.6. This may be an indication that these models show a better precision in their grounding abilities. When an object is grasped there is more certainty that the correct object was grasped. QWEN-VL achieved the most number of objects grasped, grasping an object 62% of the times, but with certainty of correctly grasping the object of around 0.5. QWEN-VL is consistent enough to ground objects, but only did so correctly on 30% of the times.

Table 19: Prompting styles tested with KOSMOS-2 for few-shot grounding.

Style	Example
Image sentence patches without instruction/answer	<code>{image_embedding} {sentence} robot.move("<phrase>{obj}</phrase><object><patch_index_{ul}><patch_index_{br}></object>, "{location}"</code>
Image sentence patches	<code>{image_embedding} Instruction: {sentence} Answer: robot.move("<phrase>{obj}</phrase><object><patch_index_{ul}><patch_index_{br}></object>", "{location}")</code>
Sentence and patches	<code>Instruction: {sentence} Answer: robot.move("<phrase>{obj}</phrase><object><patch_index_{ul}><patch_index_{br}></object>", "{location}")</code>
Sentence and fake patches	<code>Instruction: {sentence} Answer: robot.move("<phrase>{obj}</phrase><object><patch_index_ul><patch_index_br></object>", "{location}")</code>
Sentence	<code>Instruction: {sentence} Answer: robot.move("{obj}", "{location}")</code>

Table 20: Exception counts of malformed robotic API calls on the simulation per model and per query type.

Query Type	KOSMOS-2	QWEN-VL
Affordance Tray	34	2
Attribute Relative	66	12
Attribute Table	56	1
Attribute Tray	65	1
Name Relative	70	23
Name Table	49	0
Name Tray	75	0
Percentage	49.4%	4.6%

Table 21: Samples of Generated API calls that produced an exception.

Model	Generated API call
KOSMOS-2	<code>robot.move("<phrase>bananas</phrase><object> <patch_index_0211><patch_index_0311></object>", tray)</code>
KOSMOS-2	<code>robot.move("<phrase>pear</phrase><object> <patch_index_0823><patch_index_0958></object>", tray)</code>
KOSMOS-2	<code>robot.move("<phrase>tomato can</phrase></delimiter_of_multi_objects/> <patch_index_0258><patch_index_0357></object>", shelf life)</code>
KOSMOS-2	<code>robot.move("<phrase>ball that is for entertainment</phrase> <object><patch_index_0761><patch_index_0860></object>", banana)</code>
QWEN-VL	<code>robot.move("<ref>potted meat can</ref><box>(129,462) ,(421,570)</box>", "left of<ref>master chef can</ref><box>(100,469) ,(451,570)</box></code>

Table 22: Number of target bounding boxes generated on relative query types per model on the PyBullet simulation experiments.

Model	Query Type	# Target Bounding Boxes
GPT4o	Attribute Relative	120
	Name Relative	120
Gemini-1.5-flash	Attribute Relative	120
	Name Relative	120
Gemini-1.5-pro-002	Attribute Relative	117
	Name Relative	120
KOSMOS-2	Attribute Relative	0
	Name Relative	0
QWEN-VL	Attribute Relative	5
	Name Relative	3

Table 23: Number of target positions not mapped in the simulation per query type per model from the generated VGLLM responses.

Query Type	GPT4o	Gemini-1.5 flash	Gemini-1.5 pro-002	KOSMOS-2	QWEN-VL
Affordance Tray	0	2	0	86	7
Attribute Relative	0	0	3	54	65
Attribute Table	46	54	54	59	59
Attribute Tray	0	0	0	55	0
Name Relative	0	0	0	50	66
Name Table	43	50	48	62	61
Name Tray	0	0	0	45	0

5.4 Future Work

These types of models are not grounding objects robustly enough to be used in robotic applications, where different pipelines achieve much better results. These models require significant improvements in grounding capabilities and better techniques to generate robust robotic API calls need to be explored for each model, particularly when using few-shot prompting. Future research could focus on different prompting techniques such as chain-of-thought prompting and task decomposition to solve long horizon tasks. VGLLMs could be trained or fine-tuned on large-scale, grounded robotic datasets to improve their grounding accuracy and task-specific generalization. This could help achieve better results for few-shot prompting techniques, having the models trained on varied robotic data API calls from different embodiments and environments. The capabilities of VGLLMs were tested on this specific robotic environment and pipeline, but further experiments could be tested on other environments and tasks to comprehensively assess the capabilities of VGLLMs in more robotic settings. Finally, some new model versions were released while working on this thesis, SAM-2, KOSMOS-2.5 (Lv et al., 2024), (Ravi et al., 2024) and QWEN-VL 2 (P. Wang et al., 2024) which show significant improvements over the current models, can be used to replace the older versions used in this thesis.

References

- Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., ... Zeng, A. (2022). Do as i can and not as i say: Grounding language in robotic affordances. In *arxiv preprint arxiv:2204.01691*.
- Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., ... Simonyan, K. (2022). *Flamingo: a visual language model for few-shot learning*.
- Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., ... Zhou, J. (2023). *Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond*. Retrieved from <https://arxiv.org/abs/2308.12966>
- Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., ... Zitkovich, B. (2023). *Rt-2: Vision-language-action models transfer web knowledge to robotic control*. Retrieved from <https://arxiv.org/abs/2307.15818>
- Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., & Dollar, A. M. (2015). Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3), 36–52. doi: 10.1109/MRA.2015.2448951
- Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., ... Stoica, I. (2024). *Chatbot arena: An open platform for evaluating llms by human preference*. Retrieved from <https://arxiv.org/abs/2403.04132>
- Ding, M., Xiao, B., Codella, N., Luo, P., Wang, J., & Yuan, L. (2022). *Davit: Dual attention vision transformers*. Retrieved from <https://arxiv.org/abs/2204.03645>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2021). *An image is worth 16x16 words: Transformers for image recognition at scale*. Retrieved from <https://arxiv.org/abs/2010.11929>
- Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., ... Florence, P. (2023). *Palm-e: An embodied multimodal language model*. Retrieved from <https://arxiv.org/abs/2303.03378>
- Ha, H., Florence, P., & Song, S. (2023). *Scaling up and distilling down: Language-guided robot skill acquisition*. Retrieved from <https://arxiv.org/abs/2307.14535>
- He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2021). *Masked autoencoders are scalable vision learners*. Retrieved from <https://arxiv.org/abs/2111.06377>
- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). *Language models as zero-shot planners: Extracting actionable knowledge for embodied agents*. Retrieved from <https://arxiv.org/abs/2201.07207>
- Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., & McHardy, R. (2023). *Challenges and applications of large language models*. Retrieved from <https://arxiv.org/abs/2307.10169>
- Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., ... Girshick, R. (2023). *Segment anything*.

- Kumra, S., Joshi, S., & Sahin, F. (2021). *Antipodal robotic grasping using generative residual convolutional neural network*. Retrieved from <https://arxiv.org/abs/1909.04810>
- Li, J., Zhu, Y., Xu, Z., Gu, J., Zhu, M., Liu, X., ... Tang, J. (2024). *Mmro: Are multimodal llms eligible as the brain for in-home robotics?* Retrieved from <https://arxiv.org/abs/2406.19693>
- Li, Y., Liu, H., Wu, Q., Mu, F., Yang, J., Gao, J., ... Lee, Y. J. (2023). *Gligen: Open-set grounded text-to-image generation*. Retrieved from <https://arxiv.org/abs/2301.07093>
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., ... Zeng, A. (2023). *Code as policies: Language model programs for embodied control*. Retrieved from <https://arxiv.org/abs/2209.07753>
- Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). *Visual instruction tuning*. Retrieved from <https://arxiv.org/abs/2304.08485>
- Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., ... Zhang, L. (2024). *Grounding dino: Marrying dino with grounded pre-training for open-set object detection*. Retrieved from <https://arxiv.org/abs/2303.05499>
- Lv, T., Huang, Y., Chen, J., Zhao, Y., Jia, Y., Cui, L., ... Wei, F. (2024). *Kosmos-2.5: A multimodal literate model*. Retrieved from <https://arxiv.org/abs/2309.11419>
- Nasiriany, S., Xia, F., Yu, W., Xiao, T., Liang, J., Dasgupta, I., ... Ichter, B. (2024). *Pivot: Iterative visual prompting elicits actionable knowledge for vlms*.
- OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., ... Zoph, B. (2024). *Gpt-4 technical report*. Retrieved from <https://arxiv.org/abs/2303.08774>
- Oude Vrielink, J., & Kasaei, D. H. (2021). *Learning to grasp objects in highly cluttered environments using deep convolutional neural networks*. Retrieved from <https://fse.studenttheses.ub.rug.nl/id/eprint/25369>
- Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., & Wei, F. (2023). *Kosmos-2: Grounding multimodal large language models to the world*.
- Qian, S., Chen, W., Bai, M., Zhou, X., Tu, Z., & Li, L. E. (2024). *Affordancellm: Grounding affordance from vision language models*.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). *Learning transferable visual models from natural language supervision*.
- Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., ... Feichtenhofer, C. (2024). *Sam 2: Segment anything in images and videos*. Retrieved from <https://arxiv.org/abs/2408.00714>
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). *High-resolution image synthesis with latent diffusion models*. Retrieved from <https://arxiv.org/abs/2112.10752>
- Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., ... Garg, A. (2022). *Progprompt: Generating situated robot task plans using large language models*. Retrieved from <https://arxiv.org/abs/2209.11302>

- Suchi, M., Patten, T., Fischinger, D., & Vincze, M. (2019). Easylab: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets. In *International conference on robotics and automation, ICRA 2019, montreal, qc, canada, may 20-24, 2019* (pp. 6678–6684). Retrieved from <https://doi.org/10.1109/ICRA.2019.8793917> doi: 10.1109/ICRA.2019.8793917
- Tang, C., Huang, D., Ge, W., Liu, W., & Zhang, H. (2023). *Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping*. Retrieved from <https://arxiv.org/abs/2307.13204>
- Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., ... Vinyals, O. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. Retrieved from <https://arxiv.org/abs/2403.05530>
- Tziafas, G., & Kasaei, H. (2023). *Enhancing interpretability and interactivity in robot manipulation: A neurosymbolic approach*. Retrieved from <https://arxiv.org/abs/2210.00858>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2023). *Attention is all you need*. Retrieved from <https://arxiv.org/abs/1706.03762>
- Wang, H., Ma, S., Huang, S., Dong, L., Wang, W., Peng, Z., ... Wei, F. (2022). *Foundation transformers*. Retrieved from <https://arxiv.org/abs/2210.06423>
- Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., ... Lin, J. (2024). *Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution*. Retrieved from <https://arxiv.org/abs/2409.12191>
- Workshop, B., :, Scao, T. L., Fan, A., Akiki, C., Pavlick, E., ... Wolf, T. (2023). *Bloom: A 176b-parameter open-access multilingual language model*.
- Xiao, B., Wu, H., Xu, W., Dai, X., Hu, H., Lu, Y., ... Yuan, L. (2023). *Florence-2: Advancing a unified representation for a variety of vision tasks*. Retrieved from <https://arxiv.org/abs/2311.06242>
- Yang, J., Zhang, H., Li, F., Zou, X., Li, C., & Gao, J. (2023). *Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v*. Retrieved from <https://arxiv.org/abs/2310.11441>

Appendices

A Figures



(d) Detection Prompt: the running girl
Generation Prompt (modify background): The Wandering Earth

Figure 12: GroundingDINO example use with Stable Diffusion.



Figure 13: PyBullet Dataset samples.

Table 24: Object attributes from the HOTS dataset

ID	Supercategory	Category	Color	Material	Name
1	fruit	apple	red	organic	apple
2	fruit	banana	yellow	organic	banana
3	stationery	book	blue	paper	blue book
4	stationery	book	white	paper	white book
5	stationery	book	yellow	paper	yellow book
6	kitchenware	bowl	white	ceramic	bowl
7	edible	soda can	purple	aluminium	cassis can
8	edible	soda can	red	aluminium	coke can
9	edible	soda can	yellow	aluminium	fanta can
10	edible	soda can	green	aluminium	jumbo can
11	edible	soda can	blue	aluminium	pepsi can
12	kitchenware	cup	black	paper	black cup
13	kitchenware	cup	transparent	glass	glass cup
14	kitchenware	cup	red	paper	red cup
15	kitchenware	fork	black	metal	black fork
16	kitchenware	fork	silver	metal	silver fork
17	edible	juice box	green	plastic	green juice box
18	edible	juice box	orange	plastic	orange juice box
19	edible	juice box	pink	plastic	pink juice box
20	electronic	keyboard	black	plastic	keyboard
21	kitchenware	knife	silver	metal	knife
22	electronic	laptop	silver	metal	laptop
23	fruit	lemon	green	organic	lemon
24	stationery	marker	blue	plastic	blue marker
25	stationery	marker	red	plastic	red marker
26	edible	milk	white	paper	big milk
27	edible	milk	white	paper	small milk
28	electronic	monitor	black	metal	monitor
29	electronic	mouse	black	plastic	black mouse
30	electronic	mouse	silver	plastic	silver mouse
31	fruit	orange	orange	organic	orange
32	fruit	peach	yellow	organic	peach
33	fruit	pear	green	organic	pear
34	stationery	pen	black	plastic	black pen
35	stationery	pen	blue	plastic	blue pen
36	stationery	pen	red	plastic	red pen
37	kitchenware	plate	white	ceramic	big plate
38	kitchenware	plate	white	ceramic	wide plate
39	edible	pringles	black	plastic	hot pringles
40	edible	pringles	red	plastic	red pringles
41	edible	pringles	purple	plastic	purple pringles
42	stationery	scissors	black	metal	black scissors
43	stationery	scissors	silver	metal	silver scissors
44	kitchenware	spoon	blue	ceramic	blue spoon
45	kitchenware	spoon	silver	ceramic	silver spoon
46	stationery	stapler	black	metal	stapler

 Listing 3: Interleaved example of a few-shot prompt used for the Gemini models

```
[{'role': 'user', 'parts':
[<PIL.Image.Image image mode=RGB size=224x224 at 0x242931B0BB0>,
'put the mustard bottle to the tray']},
{'role': 'model', 'parts': ['robot.move((379, 459,
549, 705), "tray")']}, {'role': 'user',
'parts': [<PIL.Image.Image image mode=RGB
size=224x224 at 0x242A0E7D810>,
'relocate the chips can to the top left corner']},
{'role': 'model', 'parts': ['robot.move((544, 611, 866, 794),
"top left corner")']}, {'role': 'user', 'parts':
[<PIL.Image.Image image mode=RGB size=224x224
at 0x242A0F049D0>, 'put the tennis ball in front
of the scissors']}, {'role': 'model', 'parts':
['robot.move((812, 366, 897, 450),
(258, 285, 495, 486), "front")']}]}
```



Type: affordance
 Sentence: soda can 1
 Object: I want to drink something sweet



Type: multi_hop
 Sentence: instant noodles 1
 Object: plastic bag in front of the red cereal box



Type: attribute
 Sentence: banana 1
 Object: long, yellow fruit



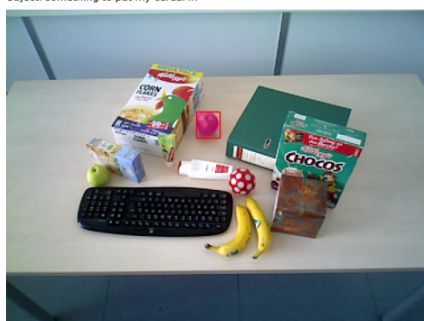
Type: name
 Sentence: food can 1
 Object: canned food



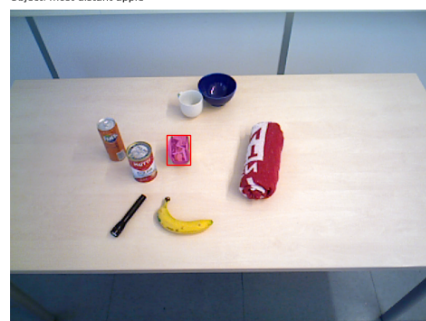
Type: semantic_relations
 Sentence: bowl 1
 Object: something to put my cereal in



Type: spatial_relations
 Sentence: apple 1
 Object: most distant apple



Type: spatial_relations
 Sentence: apple 1
 Object: most distant apple



Type: visual_relations
 Sentence: food can 1
 Object: the smallest Canned food

Figure 14: Examples of instances from the OCID dataset with the bounding box and segmentation mask of the object highlighted in red.

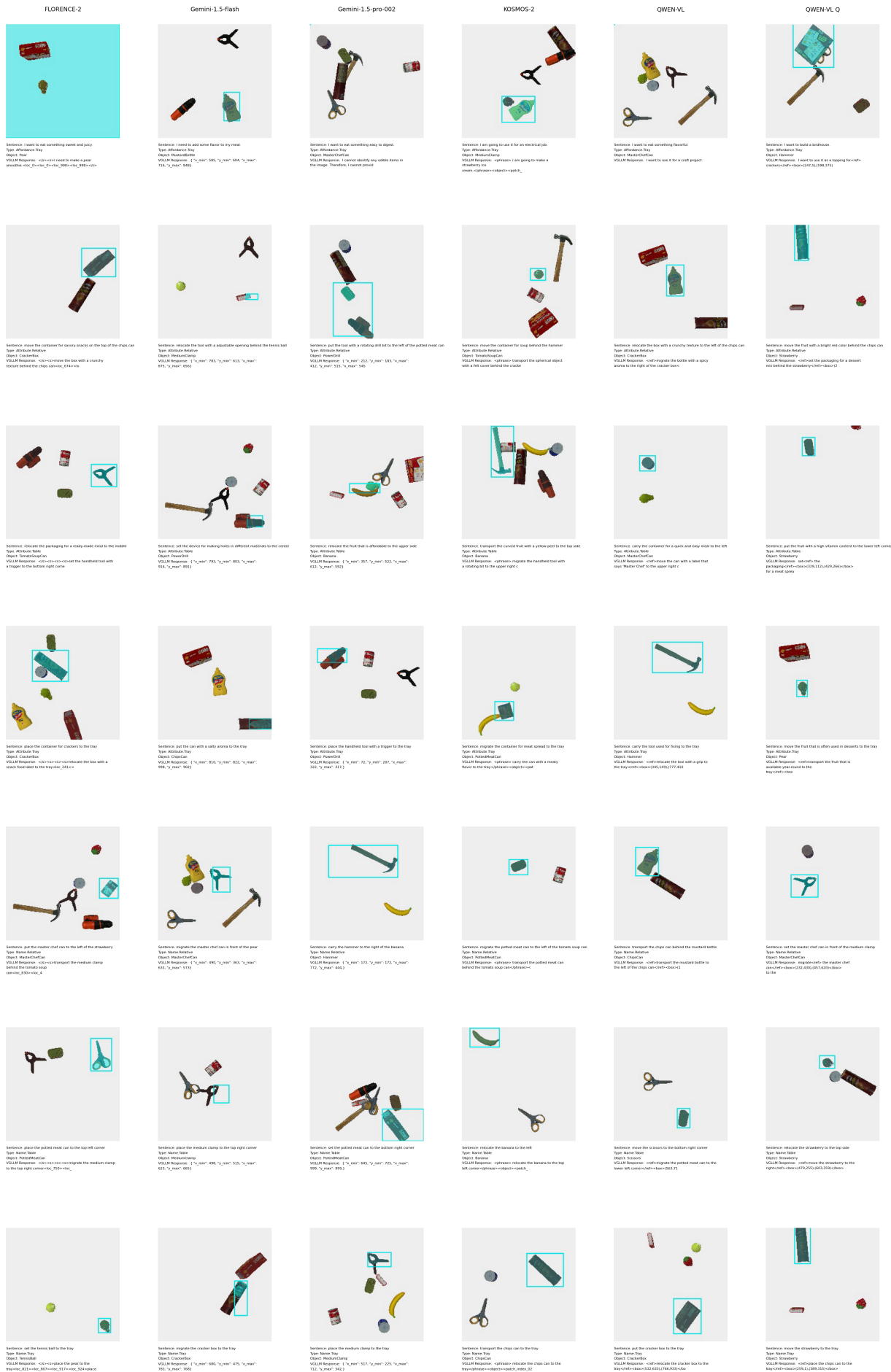


Figure 18: Random Grounding predictions samples for all models on the PyBullet dataset (Zoom in).



Figure 19: Grounding predictions samples for all models on the HOTS dataset (Zoom in).



Figure 20: Random Grounding predictions samples for all models on the HOTS dataset (Zoom in).



Figure 21: Grounding predictions for GPT4o, Gemini-1.5-flash and Gemini-1.5-pro-002 models on the Name Relative query type on the PyBullet Simulation (Zoom in).

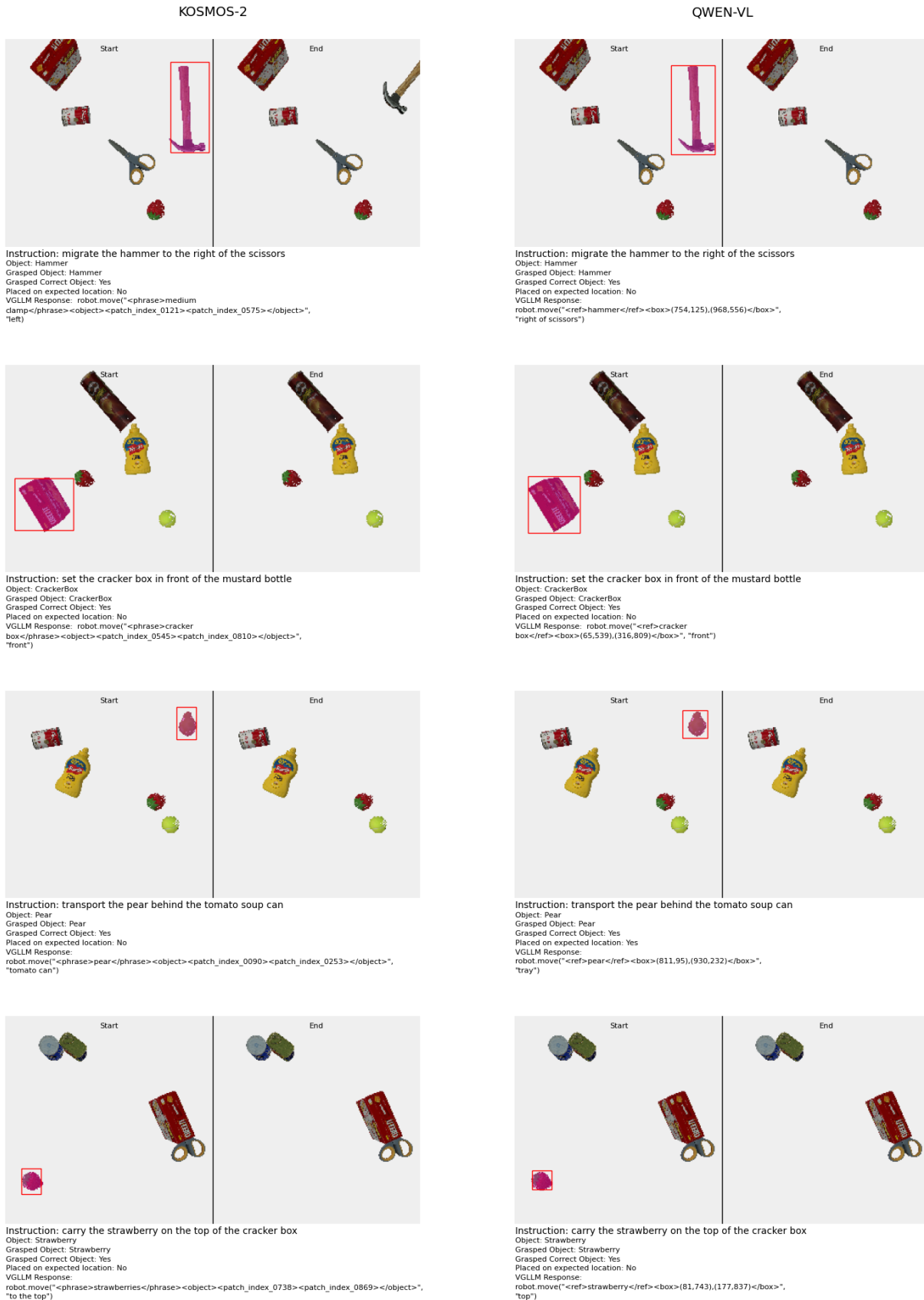


Figure 22: Grounding predictions for KOSMOS-2 and QWEN-VL models on the Name Relative query type on the PyBullet Simulation (Zoom in).



Figure 23: Grounding predictions samples for all models per query type on the PyBullet Simulation (Zoom in).

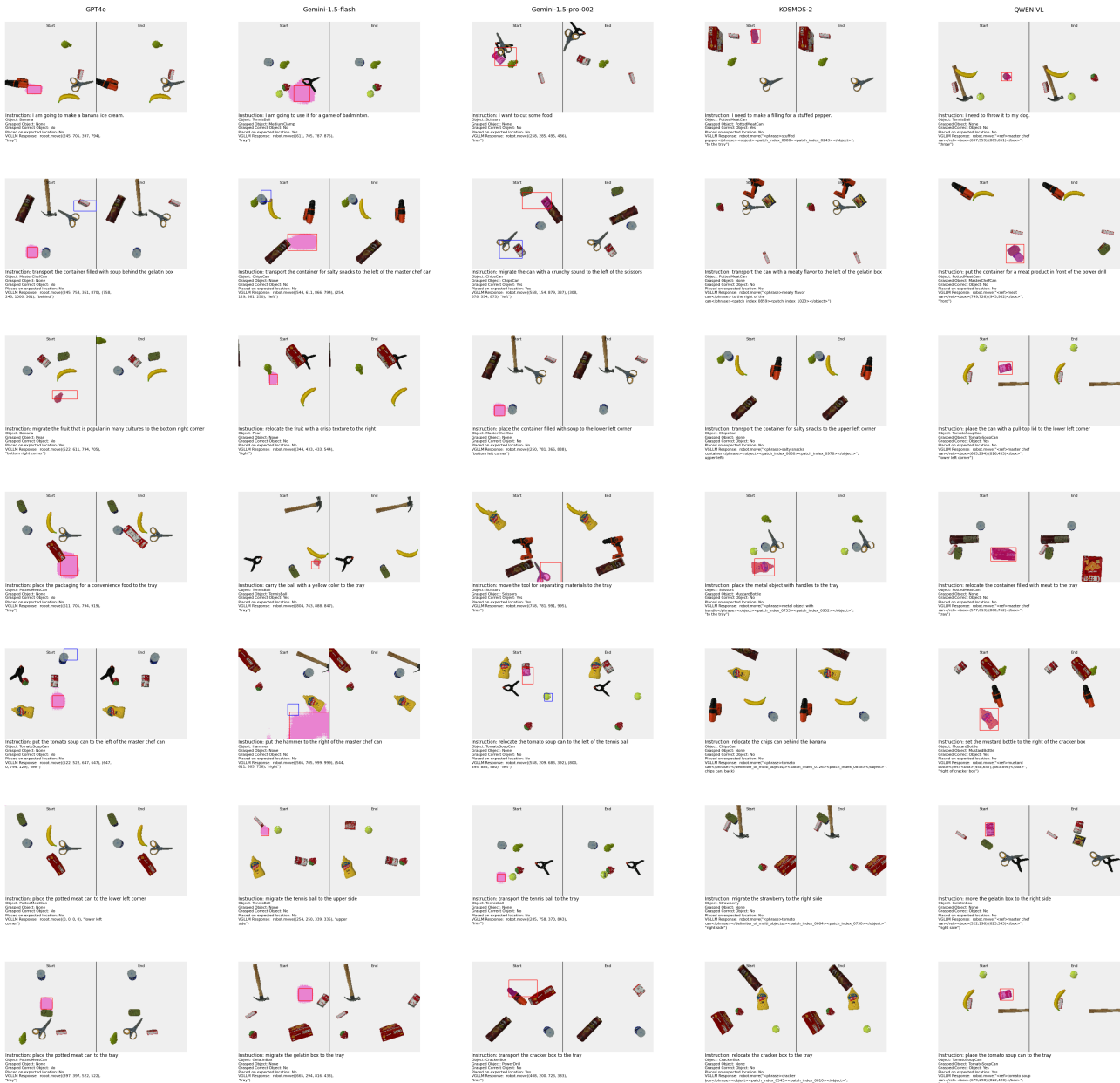


Figure 24: Random Grounding predictions for all models per query type on the PyBullet Simulation (Zoom in).

