Gondola: Grounded Vision Language Planning for Generalizable Robotic Manipulation

Shizhe Chen, Ricardo Garcia, Paul Pacaud, Cordelia Schmid
Inria, École normale supérieure, CNRS, PSL Research University
https://cshizhe.github.io/projects/robot_gondola.html

Abstract: Robotic manipulation faces a significant challenge in generalizing across unseen objects, environments and tasks specified by diverse language instructions. To improve generalization capabilities, recent research has incorporated large language models (LLMs) for planning and action execution. While promising, these methods often fall short in generating grounded plans in visual environments. Although efforts have been made to perform visual instructional tuning on LLMs for robotic manipulation, existing methods are typically constrained by single-view image input and struggle with precise object grounding. In this work, we introduce Gondola, a novel **gro**u**n**de**d** visi**o**n-**l**anguage pl**a**nning model based on LLMs for generalizable robotic manipulation. Gondola takes multi-view images and history plans to produce the next action plan with interleaved texts and segmentation masks of target objects and locations. To support the training of Gondola, we construct three types of datasets using the RLBench simulator, namely robot grounded planning, multi-view referring expression and pseudo long-horizon task datasets. Gondola outperforms the state-of-the-art LLM-based method across all four generalization levels of the GemBench dataset, including novel placements, rigid objects, articulated objects and long-horizon tasks.

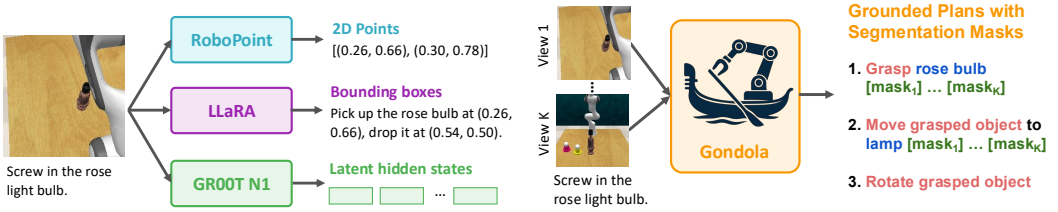
Keywords: Robotic Manipulation, Task Planning, Vision-Language Model

1 Introduction

Training robots to perform physical manipulation tasks following human instructions has been a long-term goal in robotics, enabling intuitive human-robot interaction in unstructured, dynamic environments such as homes and factories. Recently, end-to-end learning-based models [1, 2, 3], particularly Vision-Language-Action models (VLAs) [4, 5, 6, 7, 8] trained on real-world robot data, have achieved remarkable success in robotic manipulation. However, due to the scarcity and limited diversity of available robot datasets [9, 10, 11], these models still struggle to generalize beyond their training conditions, facing difficulties with novel objects, unfamiliar environments, and especially unseen long-horizon tasks [12, 13, 14].

To improve generalization, modular frameworks [15, 16, 14, 17] have received increasing attention, separating high-level task planning from low-level action execution. As the planning component is less coupled to robot embodiments, it can leverage a broader range of internet-scale data for training to enhance its generalization capabilities. Inspired by the impressive zero- and few-shot reasoning and planning abilities of Large Language Models (LLMs) [18, 19], researchers have begun exploring LLMs for task planning, such as decomposing a language instruction into substeps [14, 20] or generating executable code [15, 16]. However, since LLMs lack direct grounding in the physical world, their ability to produce actionable and reliable plans remains limited.

Different methods have been proposed to ground LLM-generated plans in visual context. SayCan [21] trains an affordance score predictor based on visual input and candidate skills, allowing the system to rerank substeps proposed by the LLM. However, this is limited to evaluating a predefined set



(a) Existing methods [24, 23, 17] using single-view input and producing various intermediate representations.

(b) Our Gondola model with multi-view inputs generating grounded plans with segmentation masks.

Figure 1: Comparison of vision-language models for high-level planning in robotic manipulation. Multi-view inputs alleviate occlusions for improved 3D scene perception, while segmentation masks offer more precise and compact grounded plans.

of skills. ECoT [22] uses image captioning models to convert visual scenes into text descriptions, which are then fed to the LLM. Yet, captions can be less accurate and miss crucial details, leading to sub-optimal decision-making and raising the risk of error propagation.

More recently, a few works [23, 24, 17] have explored fine-tuning Vision-Language Models (VLMs) to generate visually grounded plans, using intermediate representations such as points [24], bounding boxes [23], or latent hidden states [17] as illustrated in Figure 1a. While promising, points and bounding boxes are often too coarse for precise robotic manipulation in 3D environments. Latent representations from VLMs, on the other hand, are difficult to interpret and may lack the conciseness needed for efficient execution. Furthermore, most existing methods rely on single-view images, which exacerbates planning challenges due to occlusions and limited fields of view.

To address these limitations, we propose Gondola - a **gro**u**n**de**d** visi**o**n-**l**anguage pl**a**nning model to enhance generalization in robotic manipulation. As illustrated in Figure 1b, Gondola transforms language instructions and multi-view images into precisely grounded plans, consisting of interleaved actions and objects with accompanying segmentation masks for each referred object. The model builds upon a dense grounding VLM Sa2VA [25], leveraging a specialized segmentation token that enables object-specific mask generation across views. To improve planning consistency, we incorporate the textual history of previously generated plans as additional context to the model. For effectively training Gondola, we construct a robot grounded planning dataset using simulated environments from RLBench [26], supplemented with multi-view referring expression data to strengthen object grounding. To better handle long-horizon tasks, we create extended tasks by concatenating two short task sequences and using an LLM to generate instructions. We evaluate Gondola on both offline grounded planning and online task execution using the GemBench generalizable robotic manipulation benchmark [14]. Comprehensive results demonstrate Gondola's superior performance, benefiting from multi-view inputs, history-aware planning, segmentation masks and diverse training data. It outperforms the state-of-the-art LLM-based method 3D-LOTUS++ [14] by absolute 10% on average.

To summarize, our contributions are three-fold:

- We propose Gondola to generate grounded vision-language plans with masks for generalizable robotic manipulation. It features multi-view image understanding and grounding.
- We construct multi-view grounding and planning datasets using RLBench, and propose pseudo long-horizon task generation to improve long-term planning capabilities.
- Our model sets a new state of the art on the generalization benchmark GemBench, and works reliably on a real robot. The code, models and datasets will be publicly released.

2 Related Work

Vision-and-language robotic manipulation. Learning robotic manipulation conditioned on vision and language has garnered significant interest [27, 28, 29]. Due to the high-dimensional action space, directly applying reinforcement learning (RL) for training presents challenges [30]. Therefore, most approaches employ imitation learning (IL) [31, 4, 32, 33, 34, 35, 36, 37, 38] using scripted trajectories [26] or tele-operation data [9]. Visual representation plays a crucial role in policy learning. Existing works [31, 4, 2, 32, 34, 39, 40] rely on 2D images for action prediction, although recent work

explores 3D visual representations [41, 33, 35, 37, 38, 36, 14, 42]. Hiveformer [32] and RVT [34] utilize 2D images to predict a heatmap in 2D space, which is then combined with 3D point clouds to demermine the final 3D position. C2F-ARM [41] and PerAct [33] directly use 3D voxel representation as input, being less efficient due to encoding empty voxels. PolarNet [35] and 3D-LOTUS [14] improve efficiency by encoding only visible point clouds to predict actions, while SUGAR [36] further enhances point cloud representation through 3D pre-training. Given the superiority of current pre-trained 2D representations [43], works like Act3D [37] and 3D Diffuser Actor [38] lift pre-trained 2D features into 3D space, and then train 3D models to leverage the strengths of both. In this work, we leverage the strong generalization capabilities of pretrained 2D VLMs for high-level task planning and integrate it with 3D-based motion planning policies for task execution.

Foundation models for robotics. Learning-based robotic policies struggle to generalize to novel scenarios [44]. Inspired by generalization abilities of foundation models [43, 45, 19], recent research leverages these models for perception, reasoning and planning in robotics. Some methods [20, 14] directly use LLMs to decompose high-level tasks into sub-steps. To better ground plans in vision, SayCan [21] combines LLMs with value functions of pretrained skills given visual contexts. ViLa [46] replaces LLMs with a multimodal LLM GPT-4V [47]. CaP [15] directs LLMs to generate code that invokes tools for visual perception and control, and VoxPoser [16] uses LLMs and VLMs to create 3D voxel maps. These approaches rely on general-purpose pretrained models for task planning, but tend to be unstable in robotic settings and require heavy prompt engineering. To address this, a few recent methods fine-tune VLMs on robot datasets to generate grounded plans using intermediate representations such as points [24], bounding boxes [23], and latent vision-language embeddings [17]. In our work, we extend the VLM framework with multi-view inputs and finer grounding masks, and introduce synthetic robot data for long-horizon grounded planning.

Vision and language models for grounding. Early VLMs [48, 49] are constrained to generating text outputs from multimodal inputs, such as image captioning and visual question answering. To enable VLMs to produce grounded outputs that align generated texts with specific image regions, existing methods can be broadly categorized into three types. The first category outputs box coordinates [50, 51, 52, 53, 54] or polygons of segmentation masks [55] as text. However, generating precise numerical outputs is difficult and prone to hallucination. The second category uses a proposal-based approach, where a separate module first generates candidate regions, and the VLM selects the one for each generated text [56, 57]. While more structured, this method is sensitive to proposal quality, suffers from error accumulation, and introduces more computation overhead. The third category decouples language and grounding by feeding the output of a VLM into a grounding model to produce boxes or masks [58, 59, 60, 25]. Among them, Sa2VA [25] achieves the state-of-the-art performance by integrating a strong VLM model InternVL [61] and a segmentation model SAM2 [62], as well as training on large-scale image and video grounding data. Our Gondola model fine-tunes Sa2VA on multi-view image grounding and planning datasets for robotic manipulation.

3 Gondola: Grounded Vision-Language Plan Generation

We formulate high-level task planning for robotic manipulation as a vision-language grounding problem, where the goal is to generate the next executable, visually-grounded action plan that accomplishes a natural language instruction in the observed environment. Formally, given a language instruction L and multi-view visual observations $I = \{I_1, \cdots, I_K\}$ from K cameras, the Gondola model produces a grounded vision-language plan $P = (a, o, M_o, l, M_l)$, where a represents the action name, o specifies the manipulated object description paired with corresponding segmentation masks $M_o = \{m_o^1, \cdots, m_o^K\}$ across all views, and l denotes the target location description with associated location masks M_l. Noting that either o or l may be empty if the particular action does not require an object or target location for execution.

3.1 Model Architecture

As illustrated in Figure 2, the Gondola architecture consists of three main components: an image encoder for tokenizing each view image, an LLM to process multimodal inputs and outputs, and a segmentation model for multi-view object grounding.

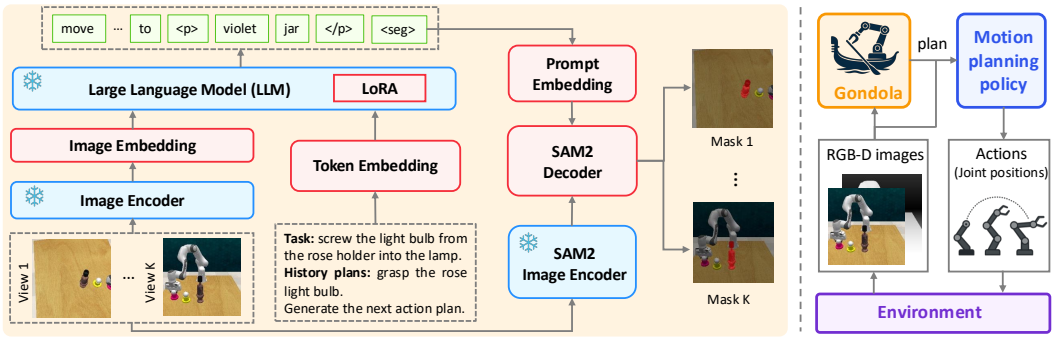


Figure 2: Left: Gondola model architecture, consisting of a shared visual encoder for multi-view images, an LLM to generate action and object names along with segmentation tokens, and SAM2 to decode masks. Right: Integrating Gondola with a motion planning policy for task execution.

Image encoder. We use a pretrained vision transformer (ViT) InternVL-300M [61] with an input image resolution of 448 $\times$ 448 to generate image patch embeddings, followed by a 2-layer multi-layer perceptron (MLP). The ViT is frozen, while the MLP is trained to adapt the visual features to the language space. The same image encoder is applied across all views, with view separation handled by a special token `\n`. Image tokens from all views are concatenated to form a single sequence.

LLM. We adopt InternVL-4B [61] as our language model, keeping its base parameters frozen while adding LoRA [63] layers for fine-tuning. Building upon Sa2VA [25], we incorporate a specialized vocabulary that includes a dedicated `<seg>` token to signal mask generation, along with paired delimiter tokens `<p>` and `</p>` that precisely delineate object and location references requiring spatial grounding. To maintain contextual awareness across sequential steps in completing manipulation tasks, we further encode previously generated history plans H as compact text tokens to the model. The following example demonstrates input and output token formatting for the LLM:

> **User**: `<image>\n<image>\n<image>\n<image>\n` You are a skilled assistant for robot task planning in tabletop environments. You can perform the following actions: grasp, move grasped object, rotate grasped object, push down, push forward, and release. Task: screw the light bulb from the rose holder into the lamp. You have completed the following action plans: grasp the rose light bulb. Please generate the next action plan.
> **Gondola**: Move the grasped object to `<p>` lamp `</p><seg>`.

Here, `<image>` represents placeholders for image tokens for each view, which are replaced by the actual visual embeddings.

Segmentation model. We employ SAM2 [62] as the segmentation model. Given the hidden embedding h_{seg} from the LLM that predicts the `<seg>` token, we project h_{seg} with a 2-layer MLP to generate a prompt embedding. SAM2 uses this prompt to segment the corresponding object mask for each view image separately and thus generates K binary masks for each referred object or location.

3.2 Training Data

We construct three datasets to train Gondola using 31 task variations in GemBench training split [14] within the RLBench simulator [26], including robot grounded planning, multi-view referring expression, and pseudo long-horizon tasks. While this data construction approach can be extended to any task in RLBench, we restrict dataset construction to the GemBench training split for fair comparison with prior work [14] in evaluating generalization performance. Figure 3 illustrates examples from each of the three datasets.

Robot grounded planning. In the RLBench simulator, each task is structured with semantically labeled objects and fixed procedure trajectories, enabling efficient grounded plan generation. First, we manually decompose the trajectory in each task into a sequence of plans, each step consisting of an action, object and placement location triplet. This only requires a single annotation effort per task with minimal annotation overhead. The corresponding segmentation masks for objects and locations are then automatically extracted given the annotated semantic labels. In this way, we create

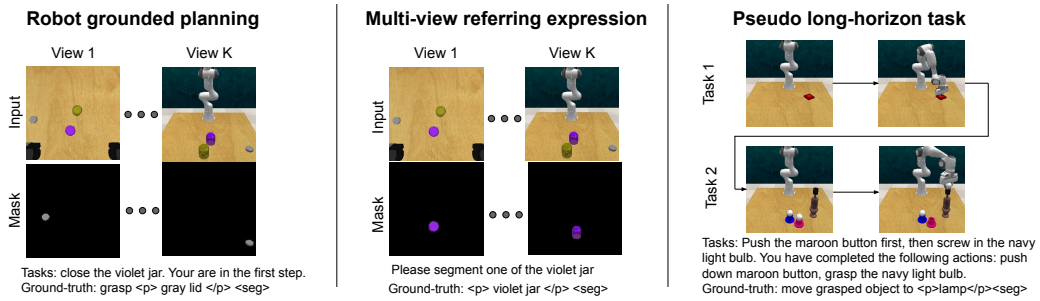


Figure 3: Constructed datasets: (1) robot grounded planning, (2) multi-view referring expressions for improved object grounding, and (3) pseudo long-horizon tasks for enhanced long-horizon planning.

ground-truth plan $\{a_t, o_t, M_o^t, l_t, M_l^t\}$ for multi-view images I_t at each keystep t [1] in an episode per task variation. We use 100 episodes for each GemBench training task variation, where each episode contains randomized object placements (and optionally new distractor objects), resulting in approximately 15k training tuples for robot grounded planning.

Multi-view referring expression. To strengthen Gondola's multi-view object grounding capabilities, we further create a multi-view referring expression dataset based on RLBench. Recognizing that the default semantic labels in RLBench contain noises and ambiguities, we implement an automatic preprocessing pipeline to standardize and refine object names in RLBench. More detail is presented in Appendix B. Similar to grounded planning generation, we automatically extract all object instances and their corresponding segmentation masks for each keystep in GemBench training split, yielding 15k multi-view image examples and 58k referring expressions. For each training example, we formulate the referring query as "Please segment one of the [object name]" with the expected output being the corresponding segmentation masks across all input view images.

Pseudo long-horizon tasks. To enhance planning for long-horizon tasks, we propose to automatically generate pseudo long-horizon sequences. Specifically, we randomly concatenate pairs of different training task sequences from GemBench training split to create compositional tasks. We then use an LLM to create coherent joint instructions for the combined tasks. Despite the abrupt scene transitions between tasks, these pseudo long-horizon tasks still help the model learn to leverage history plans to track task progress, and predict the next step based on long-horizon context.

3.3 Training Objectives

Gondola is trained to jointly optimize plan generation and multi-view object grounding. For plan generation, we employ the cross-entropy loss for next token prediction:

$$\mathcal{L}_{\text{plan}} = -\sum \log p(y_i | y_{<i}, I, L, H), \tag{1}$$

where y_i represents tokens in the generated plan including the special tokens. For multi-view object grounding, we adopt a joint loss of binary mask prediction and dice loss $\mathcal{L}_{\text{grd}} = \mathcal{L}_{\text{bce}} + \mathcal{L}_{\text{dice}}$:

$$\mathcal{L}_{\text{bce}} = -\sum [M_{\text{gt}}(p) \cdot \log(M_{\text{pred}}(p)) + (1 - M_{\text{gt}}(p)) \cdot \log(1 - M_{\text{pred}}(p))], \tag{2}$$

$$\mathcal{L}_{\text{dice}} = 1 - \frac{2 \sum_p M_{\text{pred}}(p) \cdot M_{\text{gt}}(p)}{\sum_p M_{\text{pred}}(p) + \sum_p M_{\text{gt}}(p) + \epsilon}, \tag{3}$$

where p indexes over all pixels in the mask, $M_{\text{pred}}(p)$ is the predicted probability, $M_{\text{gt}}(p)$ is the ground-truth binary label, and ϵ is a small constant for numerical stability.

4 Experiments

4.1 Evaluation Datasets and Metrics

We evaluate Gondola on the GemBench benchmark [14] for robotic manipulation in RLBench [26] simulator. GemBench assesses models' generalization capabilities across four levels: Level 1 (L1)

[1] The keystep is defined as step with significant motion change as in prior work [32, 35, 39, 14, 37], which helps avoid over-sampling similar images in training.

with new locations, Level 2 (L2) with novel rigid objects, Level 3 (L3) with new articulated objects, and Level 4 (L4) with unseen long-horizon tasks. To ensure a fair evaluation on generalization, tasks from L2 to L4 are excluded during training. The benchmark includes 31 task variations in L1, 28 in L2, 21 in L3 and 12 in L4. We conduct the following two types of evaluation:

• **Grounded planning evaluation.** This setup purely assesses models' grounded planning performance given instruction, multi-view images and ground-truth history plans. We construct a grounded planning evaluation set given the GemBench validation split. It contains 20 episodes per task variation in GemBench. For each keystep in an episode, we provide ground-truth annotations for the next action, object names and segmentation masks across all views. To evaluate the grounded planning performance, we measure the accuracy of action and object name predictions through exact text matches for each keystep. For grounding evaluation, we calculate the intersection over union (IoU) between predicted and ground-truth masks for each view. The averaged performances of each metric on all keysteps are reported for each generalization level of GemBench.

• **Task completion evaluation.** This setup integrates Gondola with low-level motion planning policies to execute the generated plans. We adopt the standard camera configuration in GemBench, using $K = 4$ cameras positioned at the front, left shoulder, right shoulder and wrist, each with an image resolution of 256×256. For evaluation, we use the GemBench test split across all four levels, conducting 20 episodes per task variation for 5 times, resulting in $20 \times 5 \times (31 + 28 + 21 + 12)$ evaluation episodes in total. Each episode is limited to a maximum of 25 steps. Task performance is measured by success rate (SR), where SR is 1 for a successful episode and 0 for failure. We report mean SR and standard deviations across the 5 runs.

4.2 Implementation Details

Training Gondola. We train the Gondola model using 8 NVIDIA H100 GPUs with training scripts built on the DeepSpeed engine [64]. The image encoder and SAM2 encoder are kept frozen during training. We apply LoRA [63] with rank 128 for parameter-efficient fine-tuning of the LLM, and the SAM2 mask decoder is also fine-tuned. The model is optimized with AdamW, using a learning rate of 2×10^{-5} for all trainable parameters. The batch size per device is set to 4, resulting in an effective batch size of 32. It takes 3 hours for training 10k iterations over the three constructed datasets.

Integrating Gondola with low-level policies. For fair comparison with prior work, we employ the same motion planning policy released by 3D-LOTUS++ [14]. Unlike 3D-LOTUS++ [14] which performs task planning only once and then executes the plan, our approach runs the task and motion planning models iteratively in a feedback loop, enabling continuous re-planning and corrections as needed. Specifically, at each step, Gondola takes multi-view RGB images, instruction and previously executed history plans as input to produce the next plan, including the next action, the manipulated object, and/or the target location together with grounded masks on each view. Then we combine aligned depth images with these masks and unify the segmented objects across views into a consolidated 3D point cloud. Following [14], each point is assigned with one of four categories based on the segmentation results and robot proprioceptive information, namely target object, target location, robot, and obstacle. The predicted action name and the point cloud are fed into the 3D motion planning policy in [14] to generate a sequence of actions. We can either run the entire action sequence as in action chunking [3] or execute one action at a time before re-planning with Gondola. We compare the two strategies in Table 3.

4.3 Ablation Studies

Boxes vs. Masks. We compare Gondola's mask-based grounding approach with a box-based variant. The box variant (row 1) in Table 1 directly generates bounding boxes as textual outputs as illustrated in the middle of Figure 1a. We use the same image encoder and LLM as Gondola (row 4) for fair comparison. To measure the mask IoU, the predicted boxes are fed into the SAM2 model to produce segmentation masks. We observe that the box-based model frequently suffers from format errors, as generating multiple numeric values for multiple images can be challenging. It performs worse across all levels in both action and object name accuracy, and shows significantly lower grounding quality in terms of mask IoU. These results highlight the advantages of end-to-end mask generation within VLMs, which provides more accurate and reliable grounding for robotic planning.

Table 1: Performance on grounded planning evaluation. We measure the action (Act) and object (Obj) name prediction accuracy and grounding performance (Grd) on the four levels of GemBench validation split. All the models are fine-tuned on the robot grounded planning dataset.

Grd type	Multi-view	Hist-ory	L1			L2			L3			L4		
			Act	Obj	Grd	Act	Obj	Grd	Act	Obj	Grd	Act	Obj	Grd
Box	✓	✓	95.1	93.7	62.8	97.4	89.0	58.7	69.3	53.8	46.2	70.0	36.8	16.6
Mask	×	×	98.0	98.2	87.8	95.1	89.5	79.8	79.3	76.3	62.7	77.2	40.0	37.4
Mask	✓	×	**100**	**100**	**88.6**	98.0	91.3	**81.2**	85.6	75.6	61.4	**89.9**	**50.1**	**46.5**
Mask	✓	✓	**100**	**100**	87.9	**99.0**	**93.3**	79.2	**88.6**	**83.9**	**66.4**	79.4	44.9	40.0

Table 2: Performance on grounded planning evaluation. All the models use multi-view and history plans for mask generation, but are fine-tuned on different composition of datasets: robot grounded planning (Plan), multi-view referring expression (RefExp), and pseudo long-horizon tasks (Long).

Finetuning Data			L1			L2			L3			L4		
Plan	RefExp	Long	Act	Obj	Grd	Act	Obj	Grd	Act	Obj	Grd	Act	Obj	Grd
✓	×	×	**100**	**100**	87.9	99.0	93.3	79.2	88.6	83.9	66.4	79.4	44.9	40.0
✓	✓	×	**100**	**100**	89.1	99.0	95.1	84.2	**92.1**	**88.2**	73.3	72.1	42.3	41.7
✓	✓	✓	**100**	**100**	**89.5**	**99.7**	**95.3**	**85.2**	88.5	82.2	**73.8**	**93.9**	**51.2**	**53.8**

Multi-view inputs. The comparison between the 2nd and 3rd rows in Table 1 showcases the impact of multi-view image inputs for robot task planning. In the 2nd row, only the front-view image is provided to the model, whereas in the 3rd row, all four views are used. Multi-view images help mitigate occlusions and generally improve action, object and grounding prediction across levels, with only slight worse performance on a few metrics in L3 compared to the single-view setting.

History plans. The last two rows in Table 1 compares the effect of incorporating history plans into task planning. Including history information boosts the performance on L2 and L3 by enabling more coherent and context-aware planning decisions. However, on L4, we observe a significant performance drop compared to the model without history. An in-depth analysis reveals that this decline is due to a distribution shift in history plans. As a result, the model tends to leverage its prior knowledge for generating purely textual plans rather than grounded plans. In contrast, the model without history does not suffer from this distribution shift. This issue can be addressed by training on our constructed pseudo long-horizon data, as shown in Table 2.

Fine-tuning datasets. Table 2 evaluates the contribution of each fine-tuning dataset. The multi-view referring expression dataset proves most effective in improving segmentation quality, leading to consistently better grounding performance across all four levels. The pseudo long-horizon task dataset is particularly beneficial for L4, as it mitigates the history plan shift issue and encourages the model to reason over extended plan histories.

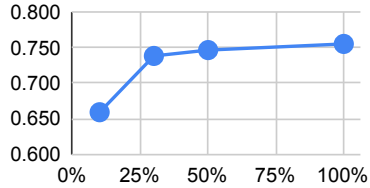


Figure 4: Averaged performance on grounded planning evaluation.

Table 3: Success rate of task execution on four levels of GemBench testing split. The Gondola model is integrated with a 3D-based motion planning policy.

Act chunk	3D filter	Uni	L1	L2	L3	L4
5	×	×	$80.8_{\pm 1.2}$	$68.5_{\pm 1.5}$	$48.6_{\pm 1.1}$	$4.1_{\pm 1.3}$
	×	✓	$87.3_{\pm 1.9}$	$74.8_{\pm 1.8}$	$\mathbf{52.4}_{\pm 2.1}$	$19.0_{\pm 1.0}$
	✓	✓	$86.5_{\pm 1.2}$	$74.4_{\pm 1.1}$	$51.1_{\pm 1.5}$	$\mathbf{19.7}_{\pm 1.7}$
1	×	✓	$\mathbf{90.8}_{\pm 1.2}$	$\mathbf{78.2}_{\pm 1.4}$	$49.5_{\pm 0.5}$	$14.9_{\pm 2.2}$
	✓	✓	$90.5_{\pm 0.3}$	$78.1_{\pm 1.8}$	$49.3_{\pm 0.9}$	$15.9_{\pm 2.1}$

Scalability and data efficiency. Figure 4 shows grounding performance averaged over 4 levels with varying percentages of finetuning data. With just 10% of the data, the model reaches 87% of the full-data performance, and performance continues to improve as more data is used.

Unified planning and grounding. Gondola unifies textual task planning and object grounding within a single framework. To highlight the benefit, we train two separate models, one for task planning and another for grounding. As shown in the first two rows of Table 3, unification yields better performance while also being more memory- and compute-efficient.

Action chunking. As shown in Table 3, when the action chunk size for running the motion planning policy is set to 1, Gondola replans at every step; when set to 5, it replans only after the motion planning policy completes the previous subplan. We observe that the impact of action chunk size

Table 4: Performance on four levels of GemBench testing split.

	Method	L1	L2	L3	L4
w/o LLM	Hiveformer [32]	$60.3_{\pm 1.5}$	$26.1_{\pm 1.4}$	$35.1_{\pm 1.7}$	$0.0_{\pm 0.0}$
	PolarNet [35]	$77.7_{\pm 0.9}$	$37.1_{\pm 1.4}$	$38.5_{\pm 1.7}$	$0.1_{\pm 0.2}$
	3D diffuser actor [38]	$91.9_{\pm 0.8}$	$43.4_{\pm 2.8}$	$37.0_{\pm 2.2}$	$0.0_{\pm 0.0}$
	RVT-2 [39]	$89.1_{\pm 0.8}$	$51.0_{\pm 2.3}$	$36.0_{\pm 2.2}$	$0.0_{\pm 0.0}$
	3D-LOTUS [14]	$\mathbf{94.3}_{\pm 1.4}$	$49.9_{\pm 2.2}$	$38.1_{\pm 1.1}$	$0.3_{\pm 0.3}$
w/ LLM	BridgeVLA [65]	$91.1_{\pm 1.1}$	$65.0_{\pm 1.3}$	$43.8_{\pm 1.2}$	$0.0_{\pm 0.0}$
	3D-LOTUS++ [14]	$68.7_{\pm 0.6}$	$64.5_{\pm 0.9}$	$41.5_{\pm 1.8}$	$17.4_{\pm 0.4}$
	Gondola (Ours)	$87.3_{\pm 1.9}$	$\mathbf{74.8}_{\pm 1.8}$	$\mathbf{52.4}_{\pm 2.1}$	$\mathbf{19.0}_{\pm 1.0}$

varies depending on tasks. Detailed results and analysis are provided in Appendix D. In general, for tasks requiring fine-grained manipulation, frequent replanning (i.e., smaller action chunks) yields better performance. In contrast, for long-horizon tasks that benefit from consistent, high-level planning, using a larger action chunk is more effective since the current Gondola model does not encode fine-grained history within subplans, which can limit coherent decision-making at this level.

3D postprocessing. We ablate a postprocessing step that applies 3D point cloud filtering using the DBSCAN algorithm to remove outlier points in grounded masks. Results show that this step offers minimal improvement, indicating that Gondola already produces spatially coherent object masks.

4.4 Comparison with state of the art

Table 4 presents a comparison of our Gondola model with state-of-the-art methods on the GemBench test split. Gondola is combined with the same motion policy in 3D-LOTUS++ [14] with action chunking size of 5 and no 3D postprocessing. The methods in the upper section do not use LLMs for planning but rely on end-to-end policy training to predict actions directly. While these methods perform well on seen tasks in L1, they show limited generalization to unseen objects in L2 and L3 and struggle with unseen long-horizon tasks in L4. BridgeVLA [65] is an end-to-end VLA model fine-tuned on the GemBench dataset. Thanks to large-scale Internet pretraining, it achieves strong performance on training tasks and generalizes better than non-pretrained models in the first block. However, its joint fine-tuning of vision and actions on the robot dataset may be suboptimal, leading to worse performance than our hierarchical model on levels L2–L4. Compared to 3D-LOTUS++ [14], which uses extensive engineering and in-context learning to enable LLM-based planning without visual input, our Gondola model offers a straightforward approach to directly generate grounded plans for follow-up motion planning. Gondola outperforms 3D-LOTUS++ [14] by 10.3% on novel rigid objects in L2, 10.9% on novel articulated objects in L3 and 1.6% in L4 of long-horizon tasks. Detailed results per task and qualitative examples are included in Appendix D. We further deploy Gondola in LIBERO and a real robot with results in Appendix E.

5 Conclusion

This paper presents Gondola, a grounded vision-language planning model to improve generalization in robotic manipulation. Gondola features with multi-view perception and the incorporation of planning history to generate fine-grained segmentation masks in the action plan. We construct three simulated datasets based on RLBench for model training, including robot grounded planning, multi-view referring expressions and pseudo long-horizon tasks datasets. Gondola demonstrates superior performance in both standalone planning and full execution on the GemBench benchmark, achieving stronger generalization abilities on novel rigid and articulated objects and long-horizon tasks. Our experiments highlight the importance of multi-view grounding, temporal reasoning, and end-to-end mask generation for effective robotic planning.

Acknowledgements. This work was partially supported by the HPC resources from GENCI-IDRIS (Grant 20XX-AD011012122 and AD011014846). It was funded in part by the French government under management of Agence Nationale de la Recherche as part of the "France 2030" program, reference ANR-23-IACL-0008 (PR[AI]RIE-PSAI projet), the ANR project VideoPredict (ANR-21-FAI1-0002-01), and the Paris Île-de-France Région in the frame of the DIM AI4IDF.

References

[1] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv:2303.04137*, 2023.

[3] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv:2212.06817*, 2022.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[7] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[8] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[9] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *CoRL*, 2023.

[10] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *RSS 2024 Workshop: Data Generation for Robotics*.

[11] AgiBot-World-Contributors, Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Hu, X. Huang, S. Jiang, Y. Jiang, C. Jing, H. Li, J. Li, C. Liu, Y. Liu, Y. Lu, J. Luo, P. Luo, Y. Mu, Y. Niu, Y. Pan, J. Pang, Y. Qiao, G. Ren, C. Ruan, J. Shan, Y. Shen, C. Shi, M. Shi, M. Shi, C. Sima, J. Song, H. Wang, W. Wang, D. Wei, C. Xie, G. Xu, J. Yan, C. Yang, L. Yang, S. Yang, M. Yao, J. Zeng, C. Zhang, Q. Zhang, B. Zhao, C. Zhao, J. Zhao, and J. Zhu. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[12] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE RA-L*, 2022.

[13] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[14] R. Garcia, S. Chen, and C. Schmid. Towards generalizable vision-language robotic manipulation: A benchmark and LLM-guided 3D policy. *ICRA*, 2025.

[15] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[16] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv:2307.05973*, 2023.

[17] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[18] AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[19] OpenAI. GPT-4 technical report. *arXiv:2302.11550*, 2023.

[20] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*, 2022.

[21] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *CoRL*, 2023.

[22] Z. Michał, C. William, P. Karl, M. Oier, F. Chelsea, and L. Sergey. Robotic control via embodied chain-of-thought reasoning. In *CORL*, 2024.

[23] X. Li, C. Mata, J. Park, K. Kahatapitiya, Y. S. Jang, J. Shang, K. Ranasinghe, R. Burgert, M. Cai, Y. J. Lee, et al. Llara: Supercharging robot learning data for vision-language policy. *arXiv preprint arXiv:2406.20095*, 2024.

[24] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[25] H. Yuan, X. Li, T. Zhang, Z. Huang, S. Xu, S. Ji, Y. Tong, L. Qi, J. Feng, and M.-H. Yang. Sa2va: Marrying sam2 with llava for dense grounded understanding of images and videos. *arXiv preprint arXiv:2501.04001*, 2025.

[26] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE RA-L*, 2020.

[27] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *IJRR*, 2021.

[28] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *IEEE RA-L*, 2023.

[29] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *NeurIPS*, 2020.

[30] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, 2018.

[31] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *CoRL*, 2022.

[32] P.-L. Guhur, S. Chen, R. Garcia Pinel, M. Tapaswi, I. Laptev, and C. Schmid. Instruction-driven history-aware policies for robotic manipulations. In *CoRL*, 2023.

[33] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2023.

[34] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *CoRL*, 2023.

[35] S. Chen, R. Garcia, C. Schmid, and I. Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. In *CoRL*, 2023.

[36] S. Chen, R. Garcia, I. Laptev, and C. Schmid. Sugar: Pre-training 3d visual representations for robotics. *CVPR*, 2024.

[37] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *CoRL*, 2023.

[38] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv:2402.10885*, 2024.

[39] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt2: Learning precise manipulation from few demonstrations. In *RSS*, 2024.

[40] G. Tziafas and H. Kasaei. Towards open-world grasping with large vision-language models. *arXiv preprint arXiv:2406.18722*, 2024.

[41] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *CVPR*, 2022.

[42] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *arXiv preprint arXiv:2409.07343*, 2024.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[44] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, D. M, J. Peralta, B. Ichter, K. Hausman, and F. Xia. Scaling robot learning with semantically imagined experience. *arXiv:2302.11550*, 2023.

[45] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[46] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv:2311.17842*, 2023.

[47] G. OpenAI. 4v (ision) system card. *preprint*, 2023.

[48] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *NeurIPS*, 2024.

[49] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024.

[50] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

[51] K. Chen, Z. Zhang, W. Zeng, R. Zhang, F. Zhu, and R. Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023.

[52] J. Chen, D. Zhu, X. Shen, X. Li, Z. Liu, P. Zhang, R. Krishnamoorthi, V. Chandra, Y. Xiong, and M. Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023.

[53] H. You, H. Zhang, Z. Gan, X. Du, B. Zhang, Z. Wang, L. Cao, S.-F. Chang, and Y. Yang. Ferret: Refer and ground anything anywhere at any granularity. *arXiv preprint arXiv:2310.07704*, 2023.

[54] W. Wang, Q. Lv, W. Yu, W. Hong, J. Qi, Y. Wang, J. Ji, Z. Yang, L. Zhao, X. Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[55] J. Liu, H. Ding, Z. Cai, Y. Zhang, R. K. Satzoda, V. Mahadevan, and R. Manmatha. Polyformer: Referring image segmentation as sequential polygon generation. In *CVPR*, 2023.

[56] C. Ma, Y. Jiang, J. Wu, Z. Yuan, and X. Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *ECCV*, 2025.

[57] Y. Zhang, Z. Ma, X. Gao, S. Shakiah, Q. Gao, and J. Chai. Groundhog: Grounding large language models to holistic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14227–14238, 2024.

[58] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia. Lisa: Reasoning segmentation via large language model. In *CVPR*, 2024.

[59] H. Zhang, H. Li, F. Li, T. Ren, X. Zou, S. Liu, S. Huang, J. Gao, C. Li, J. Yang, et al. Llava-grounding: Grounded visual chat with large multimodal models. In *ECCV*. Springer, 2025.

[60] H. Rasheed, M. Maaz, S. Shaji, A. Shaker, S. Khan, H. Cholakkal, R. M. Anwer, E. Xing, M.-H. Yang, and F. S. Khan. Glamm: Pixel grounding large multimodal model. In *CVPR*, 2024.

[61] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024.

[62] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[63] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[64] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *ACM SIGKDD*, 2020.

[65] P. Li, Y. Chen, H. Wu, X. Ma, X. Wu, Y. Huang, L. Wang, T. Kong, and T. Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *arXiv preprint arXiv:2506.07961*, 2025.

[66] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791, 2023.

A Limitations

First, data scarcity remains a major bottleneck for Gondola. Our current dataset is limited to short-horizon tasks in controlled tabletop environments from the GemBench training split, which restricts generalization to unseen objects and more complex long-horizon tasks. Expanding the dataset with more diverse simulated and real-world data is essential.

Second, visual history encoding can be improved. Multi-view inputs introduce many image tokens, making it difficult to include detailed historical context. A more efficient memory mechanism could support richer history representation without overwhelming the model.

Lastly, our approach relies solely on imitation learning from successful episodes, making it challenging for the model to anticipate and correct errors. Introducing examples of failure and recovery could better equip Gondola for robust, real-world applications.

B Data Construction in RLBench

We detail the label construction from RLBench in Section 3.2. RLBench contains scripted trajectories for each task, and every object in the scene already has a name and an associated label id. We use regular expressions to filter out undesired objects and fix object names by removing undesired name parts, prefix numbers or words such as distractor or success. For the set of objects whose color changes from one task variation to the other, we prepend the color name whose RGB value is closest to the RGB value of the object color. We consider 20 different colors that appear in RLBench: red, maroon, lime, green, blue, navy, yellow, cyan, magenta, silver, gray, orange, olive, purple, teal, azure, violet, rose, black and white. We use 3100 episodes from the 31 GemBench training task variations. For each keystep, we collect RGB images per camera viewpoint, the extracted list of object names in the scene, and object masks across views.

C Comparison with RoboPoint

RoboPoint [24] is a VLM finetuned on synthetic robot images to generate points given the instruction. Without fine-tuning on downstream data, RoboPoint performs poorly. It generates multiple points, some of which fall outside the target object, reducing overall performance. Moreover, it uses only a single-view image, resulting in inconsistency across views when multiple instances of the same category exist. In offline evaluation, RoboPoint+SAM achieves grounding IoU scores of 23.0, 22.0, 18.6, 18.2 across the four levels given ground-truth object names - significantly lower than our model.

D Detailed Results on GemBench

Table 5 to 8 present the per-task results on four levels of GemBench benchmark, respectively.

We observe that the impact of action chunk size varies depending on tasks. Frequent replanning (i.e., smaller action chunks) yields better performance for tasks requiring fine-grained and reactive actions such as 'SlideBlock' and 'PutInCupboard'. For example, in the 'PutInCupboard' task as shown in Figure 5, Step 2 involves moving the grasped object to the cupboard. Although the predictions of Gondola are correct, the partial observation of the cupboard makes it difficult to precisely localize its position for the motion planner, causing the object to fall outside the intended area. This issue could be mitigated by using smaller action chunks, which would allow the model to gradually get closer and acquire better views of the cupboard. While Gondola has replanning capabilities as shown in Step 5, the motion planner fails again due to the new object pose. Figure 6 shows a failure example for the 'SlideBlock' task. It is challenging for the motion planner to predict long-horizon action trajectories for this contact-rich task, though the initial plan generated from Gondola is correct.

In contrast, for long-horizon tasks in Level 4 that benefit from consistent, high-level planning, using a larger action chunk is more effective since the current Gondola model does not encode fine-grained history within subplans, which can limit coherent decision-making at this level.

Table 5: Detailed performance on each task variation on GemBench Level 1. 'AC' denote action chunk size.

Method	Avg.	Close Fridge+0	Close Jar+15	Close Jar+16	CloseLap-topLid+0	CloseMicro-wave+0	LightBulb In+17	LightBulb In+19	Open Box+0	Open Door+0
3D-LOTUS++ [14]	68.7	95	**100**	99	28	87	55	45	55	**79**
Gondola (AC=5)	87.3	96	99	**100**	96	82	**85**	**81**	**63**	**79**
Gondola (AC=1)	**90.8**	97	**100**	**100**	**98**	**83**	80	73	55	73

Method	Open Drawer+0	Open Drawer+2	Pick& Lift+0	Pick& Lift+2	Pick& Lift+7	PickUp Cup+11	PickUp Cup+8	PickUp Cup+9	Push Button+0	Push Button+3
3D-LOTUS++ [14]	68	75	97	94	93	91	86	88	**100**	**100**
Gondola (AC=5)	82	**97**	97	97	**98**	88	95	90	**100**	**100**
Gondola (AC=1)	**84**	96	**100**	**100**	97	**96**	**97**	**94**	97	**100**

Method	Push Button+4	PutInCup-board+0	PutInCup-board+3	PutMoney InSafe+0	PutMoney InSafe+1	Reach& Drag+14	Reach& Drag+18	Slide Block+0	Slide Block+1	Stack Blocks+30
3D-LOTUS++ [14]	**100**	1	2	22	16	94	62	**100**	65	86
Gondola (AC=5)	**100**	58	55	89	73	97	98	**100**	74	76
Gondola (AC=1)	**100**	**81**	**72**	**96**	**94**	**100**	**100**	**100**	**95**	**89**

Method	Stack Blocks+36	Stack Blocks+39
3D-LOTUS++ [14]	20	28
Gondola (AC=5)	81	81
Gondola (AC=1)	**84**	**84**

Table 6: Detailed performance on each task variation on GemBench Level 2. 'AC' denote action chunk size.

Method	Avg.	Close Jar+3	Close Jar+4	Lamp On+0	LightBulb In+1	LightBulb In+2	Pick& Lift+14	Pick& Lift+16	Pick& Lift+18	Pick&Lift Cylinder+0
3D-LOTUS++ [14]	64.5	98	96	**2**	56	43	94	96	95	**91**
Gondola (AC=5)	74.8	**99**	**100**	1	81	**83**	96	**98**	**99**	78
Gondola (AC=1)	**78.2**	**99**	99	0	**83**	73	**98**	96	98	89

Method	Pick&Lift Moon+0	Pick&Lift Star+0	Pick&Lift Toy+0	PickUp Cup+10	PickUp Cup+12	PickUp Cup+13	Push Button+13	Push Button+15	Push Button+17	PutCube InSafe+0
3D-LOTUS++ [14]	29	94	71	79	89	84	**99**	**100**	99	37
Gondola (AC=5)	**91**	**95**	77	84	95	97	**99**	99	**100**	42
Gondola (AC=1)	90	**95**	**83**	**86**	**96**	**98**	**99**	**100**	99	**57**

Method	PutInCup-board+7	PutInCup-board+8	Reach& Drag+5	Reach& Drag+7	Slide Block+2	Slide Block+3	Stack Blocks+24	Stack Blocks+27	Stack Blocks+33
3D-LOTUS++ [14]	1	0	94	64	**27**	5	22	83	59
Gondola (AC=5)	**16**	1	97	96	26	10	81	80	72
Gondola (AC=1)	15	**2**	**100**	**100**	23	**50**	**91**	**85**	**86**

Table 7: Detailed performance on each task variation on GemBench Level 3. 'AC' denote action chunk size.

Method	Avg.	Close Box+0	Close Door+0	Close Drawer+0	Close Fridge+0	Close Grill+0	CloseLaptop Lid2+0	Close Microwave2+0
3D-LOTUS++ [14]	41.5	29	**1**	**69**	93	19	50	99
Gondola (AC=5)	**52.4**	**57**	**1**	**69**	93	46	61	99
Gondola (AC=1)	49.5	51	0	46	**97**	**52**	**66**	**100**

Method	Open Box2+0	Open Door2+0	Open Drawer+1	Open Drawer2+0	Open Drawer3+0	OpenDrawer Long+0	OpenDrawer Long+1	OpenDrawer Long+2
3D-LOTUS++ [14]	16	52	0	70	41	72	52	23
Gondola (AC=5)	**20**	**57**	0	**86**	61	90	63	34
Gondola (AC=1)	10	43	0	82	**71**	**93**	**68**	**50**

Method	OpenDrawer Long+3	Open Fridge+0	OpenLaptop Lid+0	Open Microwave+0	PutMoney InSafe+2	Toilet SeatUp+0
3D-LOTUS++ [14]	78	0	86	0	13	8
Gondola (AC=5)	**93**	**6**	80	0	**58**	**26**
Gondola (AC=1)	82	3	**87**	0	16	22

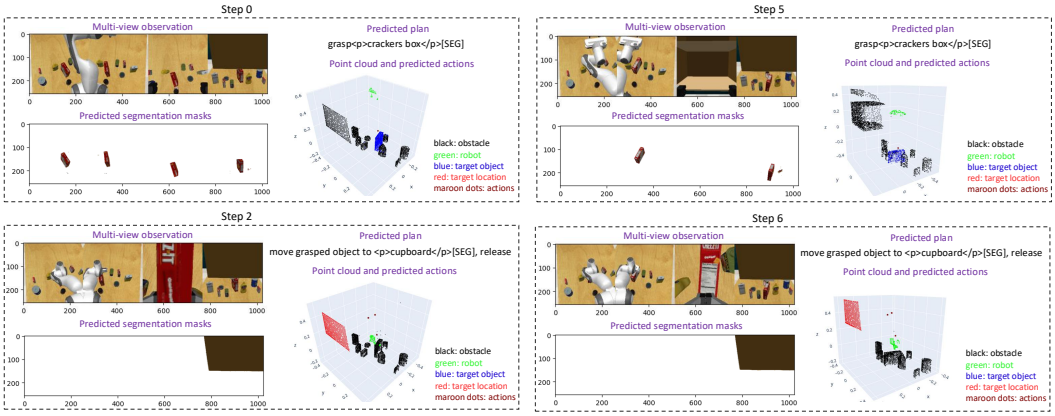


Figure 5: A failure example of the 'PutInCupboard' task using Gondola (AC=5). The instruction is 'put the crackers box in the cupboard'. The predictions of Gondola are correct, but the motion planner fails in Step 2 due to limited visual information of the cupboard from partial observations. Gondola replans in Step 5, but the motion planner fails due to the new object pose.

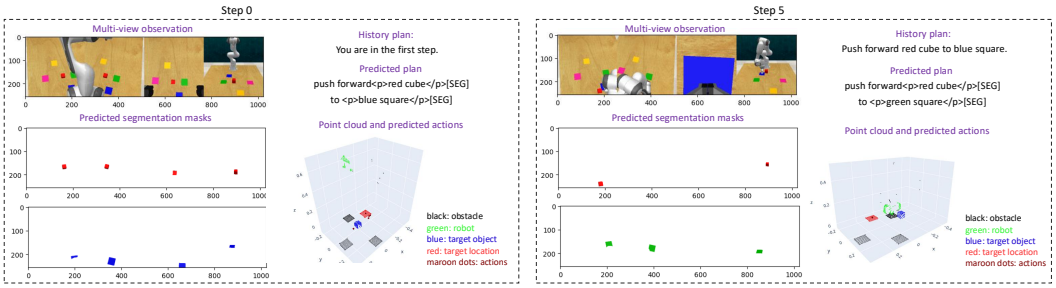


Figure 6: A failure example of the 'SlideBlock' task using Gondola (AC=5). The instruction is 'slide the block towards the blue plane'. The prediction of Gondola at Step 0 is correct, but it is challenging for the motion planner to predict long-term actions for this contact-rich task. At Step 5, due to the wrong history plan fed into Gondola, Gondola fails to replan correctly.

Table 8: Detailed performance on each task variation on GemBench Level 4. 'AC' denote action chunk size.

Method	Avg.	Push Buttons4+1	Push Buttons4+2	Push Buttons4+3	PutAllGroceries InCupboard+0	PutItems InDrawer+0	PutItems InDrawer+2	PutItems InDrawer+4
3D-LOTUS++ [14]	17.4	76	49	37	0	0	0	0
Gondola (AC=5)	**19.0**	**82**	**72**	**62**	0	0	0	0
Gondola (AC=1)	14.9	67	47	44	0	0	0	0

Method	Stack Cups+0	Stack Cup+3	TakeShoes OutOfBox+0	Tower4+1	Tower4+3
3D-LOTUS++ [14]	0	0	0	**17**	**30**
Gondola (AC=5)	0	0	0	1	11
Gondola (AC=1)	0	0	0	3	18

E Real Robot Experiments

E.1 Experimental Setup

As illustrated in Figure 7, our real robot setup includes three RealSense d435 cameras attached to a table and a 6-DoF UR5 robotic arm equipped with an RG6 gripper. We collect 20×7 demonstrations via teleoperation for 7 variations across 5 tasks: stack cup (yellow in pink or navy in yellow), put fruit (strawberry or peach) in box, open drawer, put item in drawer and hang mug. Then, we fine-tune Gondola and the 3D-LOTUS [14] motion planning policy on a joint dataset of RLBench and the real robot demonstrations. We evaluate the fine-tuned models on the same 7 seen task variations with different objects placements and evaluate generalization capabilities on 7 new unseen task variations: put fruit (lemon and banana) in box, put food (tuna can then corn) in box and put food in plates (croissant in the yellow plate and grapes in the pink plate). For each task variation, we run models 10 times and report the success rate.

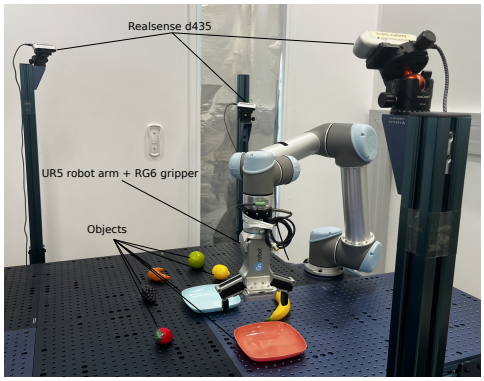


Figure 7: Our setup includes three RealSense D435 cameras and a UR5 robotics arm equipped with a RG6 gripper.

E.2 Real Robot Results

Table 9 and 10 show the performance on seen and unseen task variations, respectively. The final performance depends on both Gondola and the motion planning policy. Figure 8 illustrates a successful prediction by Gondola on a previously unseen task. However, we observe that Gondola performs worse in the real-world setting compared to simulation, primarily due to the limited amount of real robot data. As illustrated in Figure 9, the multiview consistency is significantly lower in the real world, leading to frequent segmentation errors. Increasing the availability of real-world multi-view images could help address this limitation, and we leave this direction for future work. The video in the supplementary material showcases more executions on the real robot.

Table 9: Performance of 7 seen task variations with real robot.

Task	Gondola
Stack yellow cup in pink cup	8/10
Stack navy cup in yellow cup	7/10
Put strawberry in box	4/10
Put peach in box	4/10
Open drawer	6/10
Put item in drawer	1/10
Hang mug	5/10
Avg.	**5/10**

Table 10: Performance of 7 unseen task variations with real robot.

Task	Gondola
Stack red cup in black cup	7/10
Stack black cup in orange cup	2/10
Place the yellow cup inside the red cup, then the cyan cup on top	2/10
Put lemon in box	5/10
Put banana in box	6/10
Put tuna can in box, then corn in box	3/10
Put croissant in yellow plate, then grapes in pink plate	1/10
Avg.	**3.7/10**

E.3 Qualitative Zero-shot Results on LIBERO

Figure 10 shows the zero-shot result on LIBERO benchmark [66]. Although Gondola is only finetuned on GemBench which contains different robots, numbers of cameras and environments compared to LIBERO benchmark, it still generates reasonable predictions.

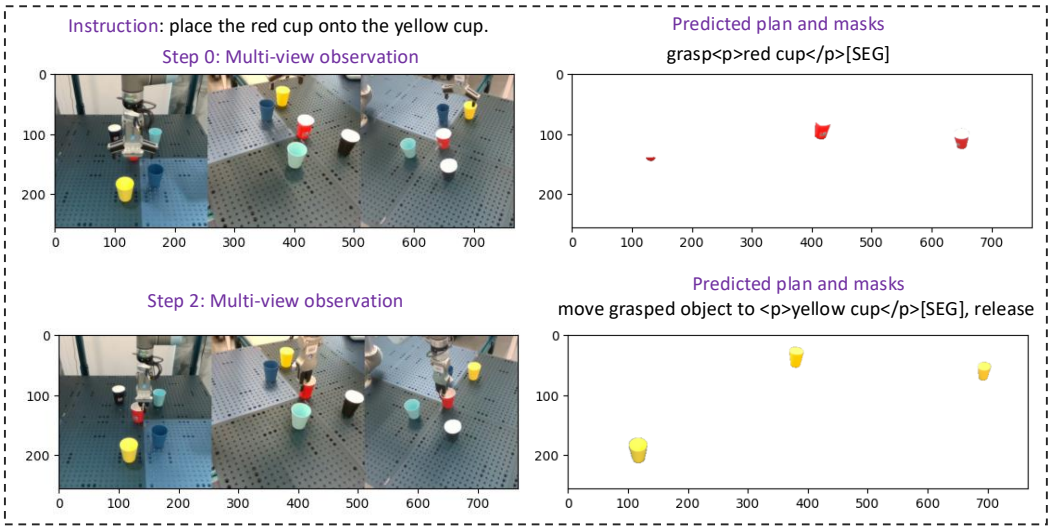


Figure 8: Successful examples of Gondola in unseen tasks with real robot.

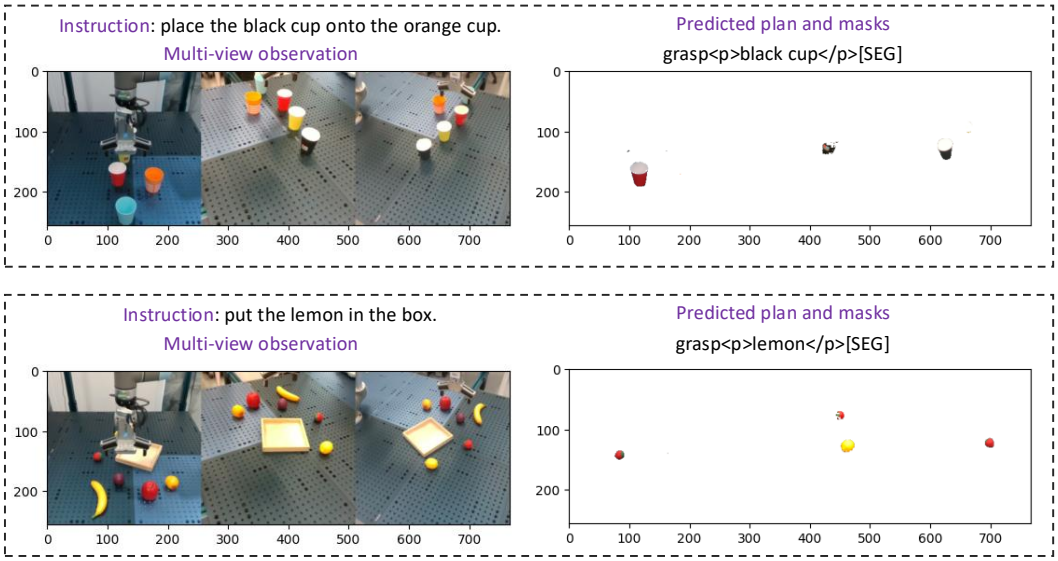


Figure 9: Failure cases of Gondola in unseen tasks with real robot.

Figure 10: Zero-shot planning result on LIBERO dataset.