2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

DoReMi: Grounding Language Model by Detecting and Recovering from Plan-Execution Misalignment

Yanjiang Guo[13*], Yen-Jen Wang[13*], Lihan Zha[2*], Jianyu Chen[13†]

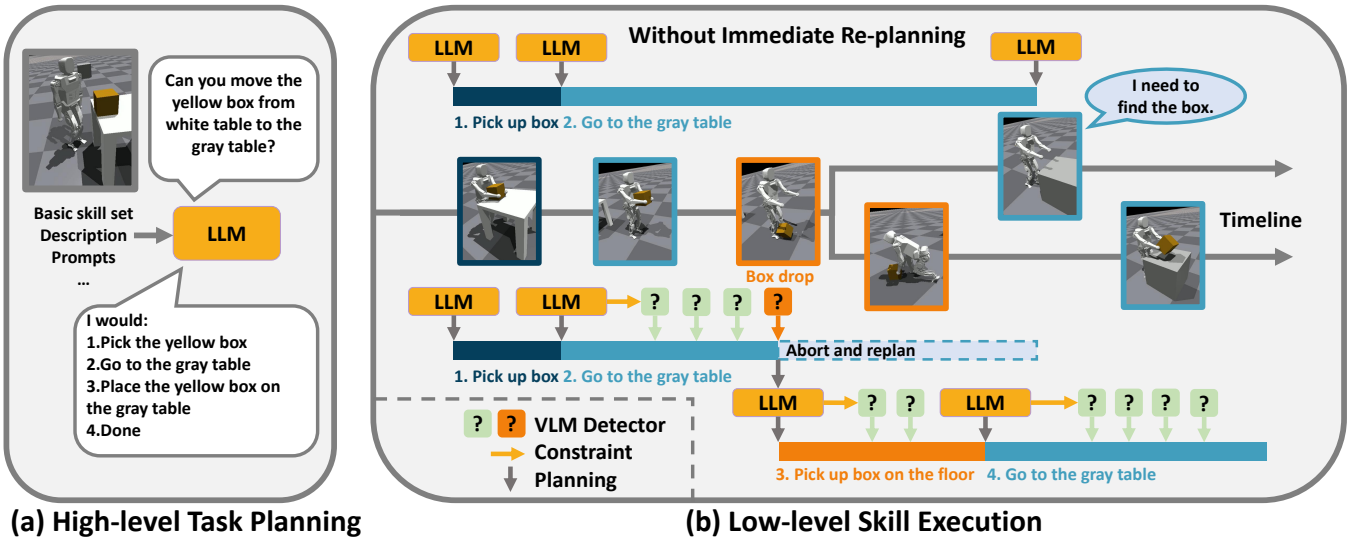


Fig. 1: Illustration of our motivation. Previous works use LLM to generate only high-level textual plans. Therefore, Low-level execution may deviate from the high-level plan. We leverage LLM to generate both plans and constraints, which enables quick recovers when misalignments happen (e.g., box drop).

***Abstract*— Large language models (LLMs) encode a vast amount of semantic knowledge and possess remarkable understanding and reasoning capabilities. Previous work has explored how to ground LLMs in robotic tasks to generate feasible and executable textual plans. However, low-level execution in the physical world may deviate from the high-level textual plan due to environmental perturbations or imperfect controller design. In this paper, we propose DoReMi, a novel language model grounding framework that enables immediate Detection and Recovery from Misalignments between plan and execution. Specifically, we leverage LLMs to play a dual role, aiding not only in high-level planning but also generating constraints that can indicate misalignment during execution. Then vision language models (VLMs) are utilized to detect constraint violations continuously. Our pipeline can monitor the low-level execution and enable timely recovery if certain plan-execution misalignment occurs. Experiments on various complex tasks including robot arms and humanoid robots demonstrate that our method can lead to higher task success rates and shorter task completion times.**

I. Introduction

Large language models (LLMs) pre-trained on web-scale data emerge with common-sense reasoning ability and understanding of the physical world. Previous works have incorporated language models into robotic tasks to help embodied agents better understand and interact with the world to complete challenging long-horizon tasks that require complex planning and reasoning [1], [2], [3].

To make the generated plan executable by embodied agents, we need to ground the language. One line of the works leverages pre-trained language models in an end-to-end manner that directly maps language and image inputs to the robot's low-level action space [4], [5], [6], [7], [8]. These approaches often require large amounts of robot action data for successful end-to-end training, which is expensive to acquire [4]. Moreover, these action-output models often contain large transformer-based architectures and cannot run at high frequencies. Therefore, they may not be suitable for tasks with complex dynamics (e.g., legged robots) that require high-frequency rapid response. Recently, many works have adopted a hierarchical approach where language models perform high-level task planning, and then some low-level controllers are adopted to generate the complex robot control commands [1], [2], [3], [9]. Under this hierarchical framework, we can leverage powerful robot control methods, such as reinforcement learning, to handle complex robot dynamic control problems with high frequency.

However, these grounding methods often assume that every low-level skill can perfectly execute the high-level plan generated by the language model. In practice, low-level execution may deviate from the high-level plan due to envi-

*Equal contribution, listed alphabetically.
†Corresponding author. jianyuchen@tsinghua.edu.cn
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. guoyj22@mails.tsinghua.edu.cn
[2]Weiyang College, Tsinghua University, Beijing, China.
[3]Shanghai Qi Zhi Institute, Shanghai, China.

ronmental perturbations or imperfect controller design. These misalignments between plan and execution may occur at any time during the task procedure. Previous works consider incorporating execution feedback into language prompts once the previous plan step is finished. If the step is unsuccessful, the process is repeated [9]. However, this delayed feedback can be inefficient. For instance, as illustrated in Figure 1(b), when a human is carrying a box and performing the low-level skill "Go to the gray table", if the box is accidentally dropped, it becomes futile to continue with the current skill. The human will immediately abort the current skill and call for the skill "Pick up the box". However, agents without immediate re-planning will continue going forward and will take more time to pick up the box dropped halfway after reaching the destination.

In this paper, we propose a novel framework **DoReMi** which enables immediate **D**etecti**o**n and **Re**covery from plan-execution **Mi**salignments. Specifically, in addition to employing LLMs for high-level planning [1], we further leverage LLMs to generate constraints for low-level execution based on their understanding of physical worlds. During the execution of low-level skills, a vision language model (VLM) [10] is employed as a general "constraint detector" to monitor whether the agent violates any constraints continuously. If some constraints are violated, indicating that the plan and execution may be misaligned, the language model is immediately called to re-plan for timely recovery. Our contributions can be summarized as follows:

- Different from previous works that use LLM only to plan, we leverage LLM to play a dual role, aiding not only in high-level planning but also generating constraints to supervise low-level execution.
- We propose DoReMi, an integrated framework between LLMs and VLMs to enable more precise and frequent feedback automatically.
- Experiments on robot arm manipulation tasks and humanoid robot tasks demonstrate that DoReMi leads to a higher task success rate and shorter task execution time.

II. Related Works

Language Grounding Prior research has attempted to employ language as task abstractions and acquired control policies that are conditioned on language [11], [12], [13], [14], [15]. Furthermore, some studies have investigated the integration of language and vision inputs within embodied tasks to directly predict the control commands [16], [17], [18]. Recent works, including [4], [5], [7], [19], [20], have demonstrated significant progress in utilizing transformer-based policies to predict actions. However, these end-to-end approaches heavily depend on the scale of expert demonstrations for model training.

Task Planning with Language Model Traditionally, task planning was solved through symbolic reasoning [21], [22] or rule-based planners [23], [24]. Recently, many works demonstrated that large language models (LLMs) can generate executable plans in a zero/few-shot manner with appropriate grounding [2], [1], [25], [26]. Some pre-trained low-level skills (primitives) are then adopted to execute steps in order. These LLM planners typically assume the successful execution of each skill, resulting in an open-loop system in physical worlds. Works in the instruction-following benchmark [27], [28] like ReAct [29], and Reflexion [30], incorporate feedback into LLM prompts to help planning after each step of the plan is finished. However, these benchmarks operate in discrete scenes and pay less attention to the skill execution period. The closest work to ours is Inner Monologue [9], which also considers continuous physical worlds, and takes into account 3 types of feedback (e.g. success detectors, scene descriptions, and human feedback) upon the completion of each step. However, Inner Monologue needs manually designed queries to get information from environments, which is impractical and hard to obtain at high frequency. In contrast to this, *our framework enables precise and high-frequency feedback with practical detectors automatically*.

Vision Language Model for Embodied Control. The vision language model (VLM) is trained on image-text pairs, enabling it to simultaneously understand visual and textual inputs and address a variety of downstream tasks, such as visual question answering (VQA)[10], [31], image captioning [32], and object detection [33]. VLMs align semantic information between vision and natural language, thereby aiding in grounding language models and facilitating embodied control. Pre-trained visual encoders or instruction encoders [34] can be connected with some action head to help train end-to-end policies [35] or generate textual plans [36]. RT-2 [5] directly fine-tuned on a VLM can generate texts and robot control actions simultaneously. VLMs can also act as scene descriptors[9], success detectors [37], [38], or object detectors[39] to facilitate the task execution. To ensure adherence to crucial constraints, we employ the VLM [40] as a "constraint detector", periodically verifying whether the agent satisfies specific constraints.

III. Method

In this section, we introduce our **DoReMi** framework which enables immediate **D**etecti**o**n and **Re**covery from Plan-Execution **Mi**salignment. Our algorithm can be succinctly described in two stages depicted in Figure 2(c):

1) At the high-level planning stage, given a set of low-level skills, prompts, and high-level task instruction, language models are leveraged to play a dual role, aiding not only in planning the next skill but also generating constraints for the next skill based on historical information.
2) During the low-level skill execution stage, we employ a vision-language model (VLM) [10] as a general "constraint detector" that periodically verifies the satisfaction of all constraints. If any constraint is violated, the language model is invoked for immediate re-planning to facilitate recovery.

A. Language Model for Planning

Following previous works that leverage LLM to generate feasible textual plans[1], we utilize LLMs to plan the next

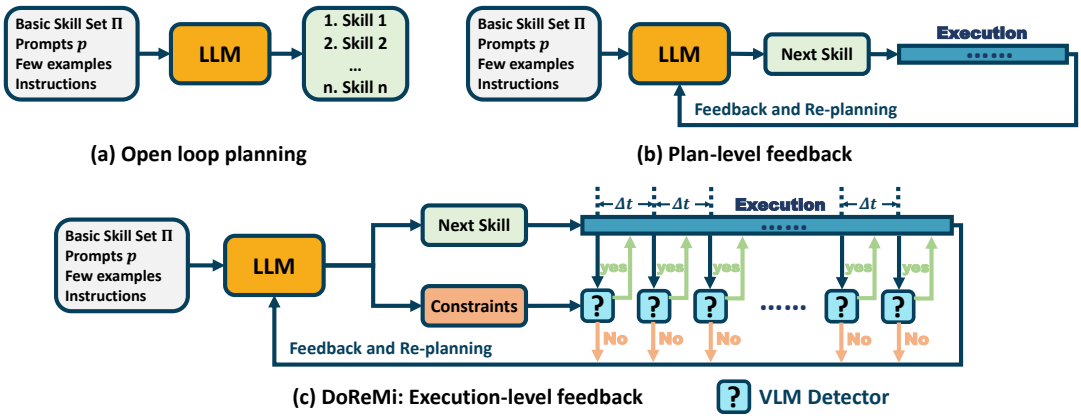
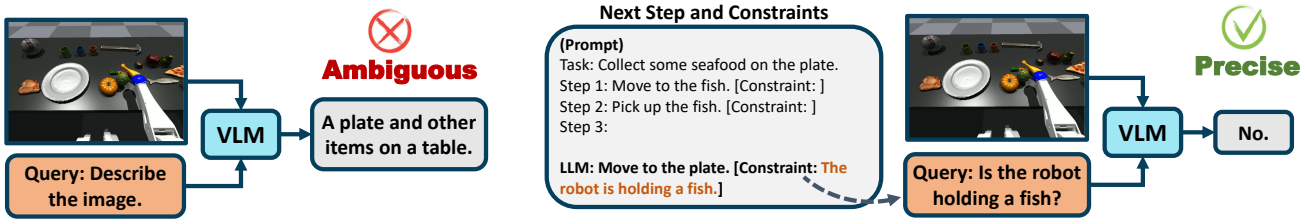


Fig. 2: Previous methods perform open-loop planning or only re-plan when the previous skill is finished. Our DoReMi framework leverages LLM to generate both the plan and corresponding constraints. Then a VLM is employed to supervise the low-level execution period, which enables immediate recovery from plan-execution misalignment.



(a) Open-ended scene description

(b) VLMs focus on specific constraint generated by LLMs

Fig. 3: Open-ended scene descriptions of VLMs are ambiguous. DoReMi leverages the LLM to reason specific constraints and actively queries the VLM for key information, resulting in much more precise feedback.

steps through few-shot in-context learning. Furthermore, we employ language models for re-planning when our constraint detector identifies a plan-execution misalignment. In such scenarios, we additionally include the misalignment information in prompts and invoke the LLM for re-planning. Practically, we deploy the Vicuna-13B model [41] locally and pick the next skill with max output probability. We also try GPT4 [42] through OpenAI API to directly output the next step with zero temperature. Both LLMs exhibit effective planning capabilities in our tasks.

B. Language Model for Constraint generation

LLM planner helps agents decompose long-horizon tasks into skill sequences. However, LLMs are not inherently integrated into the execution of low-level skills, which potentially leads to misalignment between plan and execution. To further explore the ability of LLMs in embodied tasks, we utilize LLMs not only for next-step planning but also for constraint generation based on historical information. For instance, consider the execution period of the *"go to"* skill after the *"pick up box"* skill. In such cases, the constraint *"robot holds box"* must be satisfied and violation of this constraint could indicate a failure in the picking or possible dropping of the box. Similarly, after the skill *"place red block on green block"*, the constraint *"red block on green block"* should always be met. LLMs possess the capability to automatically generate these constraints for planned steps, drawing upon their encoded understanding of the physical world. Moreover, the VLM detector can focus on these specific constraints and only needs to pick binary answers from "Yes" or "No", resulting in much more precise feedback. In contrast, open-ended scene descriptions of VLMs may result in large ambiguity and miss essential information, as shown in Figure 3.

In practice, after the LLM selects the next step with the highest output probability, we continue the generation starting with "Constraint:" to derive specific constraints.

C. VLM as Constraint Detector

Subsequent to the constraint generation stage, the agent proceeds to execute the planned step while adhering to constraints suggested by the LLM. The LLM-generated constraints may include various types, such as "red block is on blue block," "no obstacles in front of the robot," "robot is holding an apple," and more. In this work, we adopt a vision language model(VLM) [10] as a general "constraint detector" to check all constraints through visual information. The visual input of the VLM is captured from either a first-person or third-person perspective camera, and the text input is automatically adapted from the LLM proposed constraints in the form "Question: Is the *constraint* c_j *satisfied*? Answer:". For each query, the VLM only needs to select an answer from {"Yes", "No"}, which consists of very short token lengths and costs less than 0.1 second. We use $D(c_j)$ to denote the answer of the VLM D when checking constraint c_j. If c_j is satisfied, $D(c_j) = True$; otherwise, $D(c_j) = False$. The pseudo-code of the pipeline is provided in Algorithm 1. It's also worth mentioning that detectors in

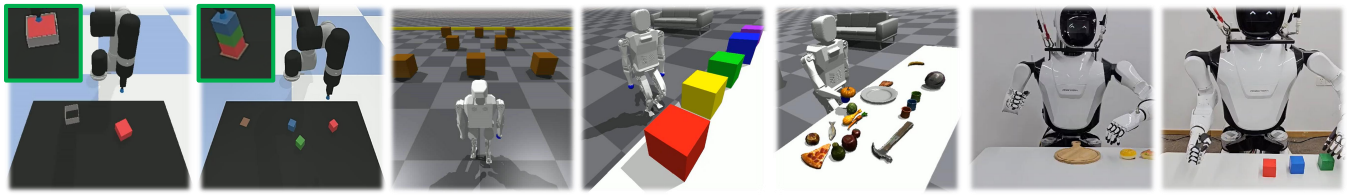


Fig. 4: Robot manipulation and humanoid robot tasks in our experiments. We consider various types of environmental disturbance and imperfect controllers in both simulation and the real world.

Algorithm 1 DoReMi (Immediate **D**etection and **Re**covery from **Mi**salignment)

Given: A high level instruction i, a skill set Π, language description l_Π for Π, language model L, prompt p_0, and VLM constraint detector D.

Initialize the skill sequence $\pi \leftarrow \emptyset$, the number of steps $n \leftarrow 1$.
while $l_{\pi_{n-1}} \neq done$ **do**
  $\pi_n \leftarrow \arg\max_{\pi \in \Pi} L(l_\pi | i, p_{n-1}, l_{\pi_{n-1}}, ..l_{\pi_0})$, $c_n \leftarrow L(i, p_{n-1}, l_{\pi_n}, ..l_{\pi_0})$
  Update prompt p_n.
  while π_n is not finished **do**
    Every Δt second, query agent all the constraints c_n using the constraint detector D.
    if $\exists D(c_n) =$ **false then**
      Add constraint violate information into prompt p_n and **break**.
    end if
  end while
  $n \leftarrow n + 1$.
end while

other modalities are also compatible with our framework and constraint detectors can run parallel to low-level controllers with different frequencies.

In practice, we use the pre-trained BLIP-2 model [10] as a general "constraint detector" to periodically check whether the agent satisfies all constraints every $\Delta t = 0.2$ second. If so, the robot continues executing the current low-level skill; otherwise, the robot aborts the current skill, and the re-planning process is triggered. We observe that pre-trained zero-shot VLM can perform well in most tasks, except those with extremely complex scenes. To enhance the performance in such complex tasks, we collect a small dataset and fine-tune the VLM using the parameter-efficient LoRA method [43]. We also verify that the fine-tuned VLM detector can generalize to unseen objects, unseen backgrounds, and even unseen tasks.

IV. Experiments

In this section, we conduct experiments involving both robotic arm manipulation tasks and humanoid robot tasks, as shown in Figure 4. These tasks incorporate various environmental disturbances and imperfect controllers, such as random dropping by the robot end-effector, noise in end-effector placement positions, failure in pick, and unexpected obstacles appearing in the robot's path.

We aim to answer the following questions: (1) Does **DoReMi** enable immediate detection and recovery from plan-execution misalignment? (2) Does **DoReMi** lead to higher task success rates and shorter task execution time under environmental disturbances or imperfect controllers?

A. Robot Arm Manipulation Tasks

Robot and Environment This environment is adapted from *Ravens* [44], a benchmark for vision-based robotic manipulation focused on pick-and-place tasks. An UR5e robot equipped with a suction gripper operates on a black tabletop, while a third-view camera provides a comprehensive view of the tabletop. The robot possesses a basic skill set including *"pick obj"* and *"place obj on receptacle"*, both of which are pre-trained primitives conditioned on single-step instructions similar to the CLIPort [35] and Transporter Nets [44]. To assess the effectiveness of our algorithm, we introduce additional disturbances into the original environment and the robot controller.

Tasks: (1) Pick and Place. The agent is required to pick a certain block and place it in a fixture. We assume the block has a probability p to drop every second when sucked by the end-effector, so the agent may need to perform pick and place several times to finish the task. **(2) Stack blocks in order.** The robot is required to stack several blocks in an order given by language instructions. The agent must perform "pick" and "place" skills in a precise sequence to successfully accomplish the task. We assume the controllers are not perfect by introducing uniform $[0, n]$ cm noise to the place positions. There is also a probability p that a block held by the end-effector might randomly drop every second. The max execution time for all tasks is set to 20 seconds. Any execution that takes time longer than 20 seconds is considered as failure.

Experiment Details Following the pipeline in Figure 2, we use Vicuna-13B [41] as LLM planner and zero-shot transferred BLIP-2 [10] as VLM constraint detector. We compare **DoReMi** with 4 baselines: **(1) SayCan**: an LLM is utilized to decompose instructions into steps and execute them sequentially. However, this approach assumes the successful execution of each step without considering potential failures. **(2) CLIPort**: a multi-task CLIPort policy conditioned on the single pick-place step. It utilizes an LLM to decompose instructions into steps and repeat each step until success. The same VLM is leveraged as a success

Tasks with disturbance		Success Rate(%) ↑					Execution Time(s) ↓		
		SayCan	CLIPort	IM	DoReMi (ours)	IM-Oracle	IM	DoReMi (ours)	IM-Oracle
Pick and place with random drop p	p=0.0	100 (±0)	100 (±0)	100 (±0)	100 (±0)	100 (±0)	**2.7 (±0.0)**	**2.7 (±0.0)**	2.7 (±0.0)
	p=0.2	81 (±9)	100 (±0)	100 (±0)	100 (±0)	100 (±0)	3.4 (±0.2)	**3.0 (±0.2)**	3.4 (±0.2)
	p=0.3	63 (±9)	100 (±0)	100 (±0)	100 (±0)	100 (±0)	4.0 (±0.2)	**3.3 (±0.2)**	4.0 (±0.2)
Stack in order with noise n	n=0.0	**100 (±0)**	**100 (±0)**	**100 (±0)**	**100 (±0)**	100 (±0)	**7.2 (±0.0)**	**7.2 (±0.0)**	7.2 (±0.0)
	n=1.0	96 (±4)	96 (±4)	96 (±4)	**100 (±0)**	100 (±0)	8.0 (±3.0)	**7.5 (±0.5)**	7.4 (±0.5)
	n=2.0	63 (±9)	85 (±7)	86 (±7)	**96 (±4)**	98 (±2)	12.2 (±5.3)	10.2 (±1.7)	9.8 (±2.0)
	n=3.0	31 (±11)	74 (±10)	75 (±8)	**86 (±8)**	91 (±7)	-	15.6 (±3.2)	14.7 (±2.3)
Stack in order with noise n random drop p=0.1	n=0.0	71 (±9)	94 (±7)	94 (±6)	**98 (±4)**	99 (±1)	10.0 (±3.6)	**9.4 (±1.7)**	9.9 (±1.9)
	n=1.0	71 (±9)	**94 (±7)**	**94 (±7)**	**94 (±7)**	97 (±2)	10.7 (±3.9)	**10.6 (±3.2)**	10.9 (±3.0)
	n=2.0	54 (±12)	79 (±9)	79 (±8)	**92 (±6)**	95 (±3)	-	**14.5 (±3.4)**	15.3 (±3.5)
	n=3.0	21 (±9)	33 (±10)	34 (±10)	**55 (±10)**	64 (±8)	-	-	-

TABLE I: Success rates and task execution time under different degrees of disturbances. We only measure execution time under high success rates. The results show the mean and standard deviation over 4 different seeds, each with 12 episodes.

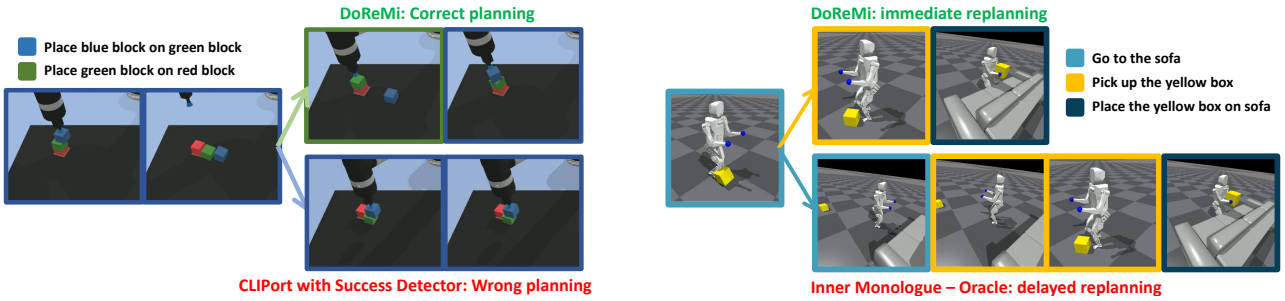


Fig. 5: Comparison examples with baselines. In the left figure, the blocks collapse during manipulation. DoReMi detects this misalignment and replans to pick and place the green block first while the baseline continues to repeat the previous step and results in failure. In the right figure, the box dropped during transportation, DoReMi immediately detects the violation and replan, which is more efficient than baseline.

detector to determine whether the current step should be repeated. **(3) Inner Monologue (IM)**: The same VLM is employed as scene descriptors and success detector to help LLM re-plan upon completion of each step. **(4) IM-Oracle:** Inner-Monologue with oracle feedback which does not exist in practical real-world settings. Results are shown in Table I.

Result Analyses In the presence of disturbances, SayCan consistently fails in all tasks due to its lack of success detectors and re-planning mechanisms. In simple pick-place tasks, CLIPort and Inner-monologue with success detector can repeat the step and recover. However, they do not have a mechanism to abort the current execution and only re-plan at the end of each skill, resulting in a longer execution time. In the stack-block task, when encountering situations that require re-planning (e.g., the blocks collapse), CLIPort that only repeats the previous step fails to recover, as shown in Figure 5. When provided with imperfect scene descriptors (VLM), Inner Monologue also struggles to recover due to ambiguous open-ended scene descriptions. In contrast, DoReMi leverages LLMs to propose specific constraints for every low-level skill, with the VLM focused on these constraints, leading to highly accurate feedback. Furthermore, our VLM continuously detects constraint violations throughout the execution period, which enables immediate re-planning and recovery. Under these two mechanisms, DoReMi reaches higher success rates and shorter execution times.

B. Humanoid Robot Tasks

Robot Description and Low-level Skill Set The humanoid robot utilized in our experiments possesses 6 degrees of freedom per leg and 4 degrees of freedom per arm, totaling 20 degrees of freedom. Controlling complex humanoid robots with a single policy is challenging. Following the framework in [45], we employ reinforcement learning to train the locomotion policy and leverage model-based controllers to acquire the manipulation policy. Specifically, we utilize the Deepmimic algorithm [46] to train a locomotion policy conditioned on commanded linear and angular velocity, allowing the robot to execute low-level skills such as "go forward 10 meters," "move forward at speed v," "go to *target place*," "turn right/left," and more. As for the manipulation policy, in simulation, we introduce an assistant pick-primitive similar to [47]; In the real world, we use dexterous hands with factory-designed pick primitives. These setups allow the robot to execute low-level skills like "pick up *object*" and "place *object* on *receptacle*".

Real world robot setup The real humanoid robot is equipped with the basic low-level controllers listed above. The LLMs and VLMs are running on the cloud and communicate with the robot at 1Hz.

We consider 5 categories of tasks:

1) **Obstacle-avoidance.** The robot is commanded to reach a finish line with unknown obstacles appearing on the way with density d. Therefore, the robot needs to satisfy the constraint *"no obstacle in the front"*. If the constraint is violated, it must perform skill *"turn left/right"* to avoid the collision.

Tasks with disturbance		Success Rate(%) ↑ SayCan	IM	DoReMi (ours)	DoReMi-FT (ours)	IM-Oracle	Execution Time(s) ↓ DoReMi-FT (ours)	IM-Oracle
Obstacle-avoidance with density d	**d=0.0**	**100 (±0)**	**100 (±0)**	**100 (±0)**	**100 (±0)**	100 (±0)	**24.2 (±0.8)**	24.2 (±0.8)
	d=0.3	68 (±6)	68 (±6)	**92 (±6)**	**92 (±6)**	68 (±6)	**31.2 (±2.4)**	-
	d=0.6	40 (±8)	40 (±8)	**90 (±6)**	**90 (±6)**	40 (±8)	**34.3 (±3.2)**	-
Move-box with random drop p	**p=0.0**	**98 (±2)**	**98 (±2)**	**97 (±2)**	**97 (±2)**	98 (±2)	**32.2 (±2.5)**	32.1 (±2.5)
	p=0.02	61 (±7)	63 (±7)	**95 (±4)**	**96 (±4)**	98 (±2)	**35.0 (±3.0)**	46.5 (±4.7)
	p=0.04	42 (±9)	46 (±9)	**94 (±4)**	**96 (±4)**	96 (±2)	**37.3 (±3.1)**	61.2 (±7.6)
Prepare-food with pick failure p_1=0.1 random drop p	**p=0.0**	78 (±5)	83 (±4)	85 (±6)	**96 (±3)**	99 (±1)	**27.6 (±2.7)**	27.8 (±3.0)
	p=0.02	49 (±5)	56 (±5)	66 (±4)	**93 (±5)**	97 (±2)	**31.0 (±3.8)**	36.8 (±5.8)
	p=0.04	18 (±5)	21 (±7)	37 (±8)	**91 (±6)**	96 (±2)	**35.2 (±6.5)**	46.3 (±7.5)
Prepare-food-real		-	20	20	**90**	-	**195.0**	-
Stack-real		-	10	20	**80**	-	**240.0**	-

TABLE II: Success rates and task execution time under different degrees of disturbances. We only evaluate execution time under high task success rates.

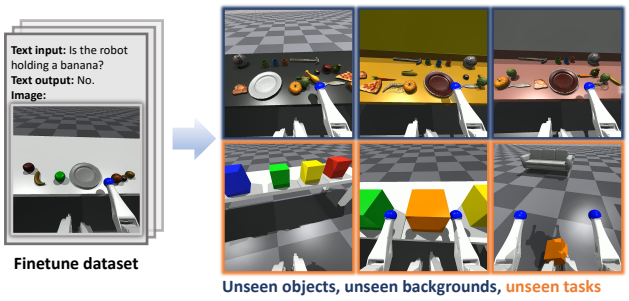


Fig. 6: VLM detector fine-tuned on the small dataset can generalize to unseen objects, unseen background, and even unseen tasks.

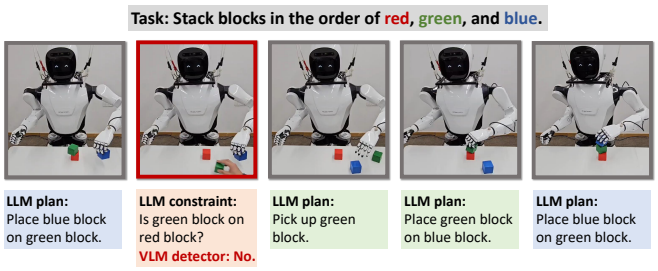


Fig. 7: Stacking task on real humanoid robot. During the step "place blue block on green block", green block was knocked down by external forces. DoReMi-FT successfully recovered from this misalignment

2) **Move-box.** The robot is required to transport a certain box from one location to another. A proper solution might involve 1) Go to place A. 2) Pick up box. 3) Go to place B. 4) Put down box. We introduced additional perturbations to this task by assuming that the robot has a probability p of dropping the box every second during transport.
3) **Prepare-food.** The robot is required to collect 2-5 types of foods from random positions according to abstract language instructions with pick failure probability p_1 and drop probability p (example in Figure 3b).
4) **Prepare-food-real** This pick-place experiment is performed on a real humanoid robot. We add external disturbances to knock off the carried object.
5) **Stack-real** A Real humanoid robot is required to stack the block in a certain order. Blocks may collapse under external forces or place noise. An example is shown in Figure 7.

Baselines Following the pipeline in Figure 2(c), we use Vicuna-13B [41] as the LLM planner and BLIP-2 [10] as the VLM constraint detector. Additionally, we use **DoReMi-FT** to denote DoReMi with fine-tuned VLM. We compare our methods with (1) **SayCan** [1], (2) **Inner Monologue (IM)** [9], (3) **IM-Oracle**: Inner monologue with Oracle perfect feedback. Since oracle feedback does not exist in practical real-world settings, we just use this baseline to reflect the upper bound of the planning performance.

Result Analyses The results are shown in Table II. Similar to analysis in section IV-A, SayCan failed due to the absence of re-planning mechanisms and Inner-monologue failed because of the ambiguity and the low frequency of the feedback. Furthermore, DoReMi-FT even surpasses IM-oracle in execution time while maintaining similar success rates due to its immediate detection and recovery mechanism, as depicted in the right of Figure 5.

In our experiments, we observed that the performance of zero-shot transferred VLM diminishes as the scene complexity increases, such as in the prepare-food(-real) task involving multiple objects. In order to enhance the performance in complex scenarios, we collected a small dataset to fine-tune the pre-trained BLIP-2 VLM [10]. The dataset only consisted of 128 image-text pairs with 5 demonstrations on fruit objects. In both simulated and real-world experiments, we are delighted to find that DoReMi-FT with the fine-tuned BLIP-2 model can generalize to complicated scenes with unseen objects, unseen backgrounds, and even unseen tasks. As shown in Figure 6, test tasks include new categories of objects like junk food, vegetables, and seafood with random positions, as well as unseen backgrounds. Detailed

information on the finetune process can be found in the Appendix.

V. Conclusion

When employing language models for embodied tasks in a hierarchical approach, the low-level execution might deviate from the high-level plan. We emphasized the importance of continuously aligning the plan with execution and leveraged LLM to generate both plan and constraints, which enables grounding language through immediate detection and recovery under practical VLM constraint detectors. Variety of challenging tasks in disturbed environments demonstrated the effectiveness of DoReMi.

on the last page of the document manually. It shortens the textheight of the last page by a suitable amount. This command does not take effect until the next page so it should come on the page before the last. Make sure that you do not shorten the textheight too much.

APPENDIX

A. Effectiveness of VLM finetuning

After fine-tuning, the accuracy of VLM in the prepare-food task has significantly increased, as shown in Table III. We use the LoRA (Low-Rank adaptation)[43] method to finetune the BLIP-2 Flan-T5-xl model, the whole training process is finished on a single Nvidia A100 card.

	Before finetune				After finetune			
	TP	FN	FP	TN	TP	FN	FP	TN
Obstacle	120	5	0	14	121	4	0	14
Move box	140	0	6	22	140	0	2	26
Prepare food	78	27	8	25	99	6	1	32

TABLE III: Accuracy analysis of VLM on humanoid tasks.

B. VLM ablation study

We also conduct an ablation study on different types of VLM with max disturbances in our tasks, as shown in Table IV. The BLIP-2 model performs similarly to the Instruct-BLIP model. However, all zero-shot transferred models can not perform well in complicated prepare-food tasks.

Success rate%	BLIP-1	BLIP-2	Instruct-BLIP
Obstacle-avoidance	88	90	92
Move-box	64	94	92
Prepare-food	16	37	40

TABLE IV: Ablation study on zero-shot transferred VLM

C. Detalied pipeline

Example for stack-in-order task:

LLM input: The robot performs manipulation tasks. At the same time, the robot needs to satisfy some constraints to ensure the successful execution of each task. Just fill in the blank and directly output the next step. Task: Stack blocks in the order of brown, red, and green.

LLM output step by step: (1) Pick the red block, [Constraint: The robot is holding red block],

(2) Place the red block on the brown block, [Constraint: The red block is on the brown block],

(3) Pick the green block, [Constraint: The robot is holding green block, the red block is on the brown block],

(4) Place the green block on the red block, [Constraint: The red block is on the brown block, the green block is on the red block], **[Detector feedback: the red block is not on the brown block]**,

(5) Pick the green block, [Constraint: The robot is holding green block, the red block is on the brown block],

(6) Place the red block on the brown block, [Constraint: The red block is on the brown block],

(7) Pick the green block, [Constraint: The robot is holding green block, The red block is on the brown block],

(8) Place the green block on the red block, [Constraint: The red block is on the brown block, The green block is on the red block]

References

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[2] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[3] 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," *arXiv preprint arXiv:2209.07753*, 2022.

[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 preprint 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] 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 *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[7] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[8] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, *et al.*, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Conference on Robot Learning*. PMLR, 2022, pp. 1303–1315.

[9] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[10] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[11] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: Connecting language, knowledge, and action in route instructions," *Def*, vol. 2, no. 6, p. 4, 2006.

[12] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[13] Y. Jiang, S. S. Gu, K. P. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[14] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," *arXiv preprint arXiv:1704.08795*, 2017.

[15] H. Mei, M. Bansal, and M. Walter, "Listen, attend, and walk: Neural mapping of navigational instructions to action sequences," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[16] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay, "Lancon-learn: Learning with language to enable generalization in multi-task manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1635–1642, 2021.

[17] P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid, "Instruction-driven history-aware policies for robotic manipulations," in *Conference on Robot Learning*. PMLR, 2023, pp. 175–187.

[18] P. Goyal, S. Niekum, and R. Mooney, "Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards," in *Conference on Robot Learning*. PMLR, 2021, pp. 485–497.

[19] Y. Zhang and J. Chai, "Hierarchical task learning from language instructions with unified transformers and self-monitoring," *arXiv preprint arXiv:2106.03427*, 2021.

[20] 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," *arXiv preprint arXiv:2210.06407*, 2022.

[21] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila, "Shop: Simple hierarchical ordered planner," in *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, 1999, pp. 968–973.

[22] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[23] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[24] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, pp. 363–373, 2019.

[25] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[26] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *arXiv preprint arXiv:2307.01928*, 2023.

[27] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[28] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[29] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[30] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *arXiv preprint arXiv:2303.11366*, 2023.

[31] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[32] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and vqa," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 13 041–13 049.

[33] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[34] 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 *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[35] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[36] 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.

[37] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," *arXiv preprint arXiv:2303.07280*, 2023.

[38] X. Zhang, Y. Ding, S. Amiri, H. Yang, A. Kaminski, C. Esselink, and S. Zhang, "Grounding classical task planners via vision-language models," *arXiv preprint arXiv:2304.08587*, 2023.

[39] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, *et al.*, "Open-world object manipulation using pre-trained vision-language models," *arXiv preprint arXiv:2303.00905*, 2023.

[40] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 888–12 900.

[41] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[42] OpenAI, "Gpt-4 technical report," *arXiv*, 2023.

[43] 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.

[44] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," *Conference on Robot Learning (CoRL)*, 2020.

[45] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022.

[46] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[47] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.