Apendix for REMAC: Self-Reflective and Self-Evolving Multi-Agent Collaboration for Long-Horizon Robot Manipulation

Anonymous submission

Project Code

Code — github.com/reamac-repo/remac-repo

Training Details

Dataset Format

The dataset used in this paper to train the diffusion policy for primitive actions was generated using the **MimicGen** (Mandlekar et al. 2023) tool. The format of this dataset follows the standard from the **RoboCasa** (Nasiriany et al. 2024) research and is based on the `robomimic` `.hdf5` file convention.

The core structure of each `.hdf5` dataset file is organized as follows:

- `data`: This group contains the environment metadata and all demonstration data.
  - `env_args` (attribute): Records metadata information about the task environment.
  - `demo_<n>`: Stores all data for demonstration *n*.
    - `model_file` (attribute): The XML string corresponding to the MuJoCo MJCF model.
    - `ep_meta` (attribute): Metadata for a single episode, such as the natural language description of the task, scene information, object information, etc.
    - `actions`: Environment actions, ordered by time. The shape is (N, A), where N is the length of the trajectory, and A is the dimension of the action space.
    - `action_dict`: A dictionary that splits actions by fine-grained components (e.g., end-effector position, rotation, gripper state, etc.).
    - `obs`: A dictionary containing various observation data, such as images, proprioception, etc.
    - `states`: Raw, low-level MuJoCo state vectors, ordered by time. This data is used only for replaying demonstrations and is **not to be used for policy learning**.
- `mask`: This group contains filter key metadata used to split the dataset in different ways (e.g., by specific skills or outcomes).

Training Method

Our experimental setup follows the standard configuration used in the **robomimic**(Mandlekar et al. 2021) framework.

Model Architecture. Our policy network processes multimodal inputs, including visual and proprioceptive state observations. For visual processing, we employ a ResNet-18 backbone that encodes 84×84 pixel images into a 64-dimensional feature vector, using a Spatial Softmax layer for pooling. For the policy itself, the actor's output head is a Gaussian Mixture Model. The actor network is an MLP with two hidden layers of size 400 and 300, respectively, using ReLU activation functions.

Training Hyperparameters. We train our model using a Behavior Cloning objective from the robomimic suite. We use the AdamW optimizer with an initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-5}. The learning rate is decayed using a cosine annealing scheduler. The model is trained for 2000 epochs with a batch size of 256.

Pre- and Post-Check Module

In this section, we elaborate on the composition of pre- and post-conditions check module, and the former includes multiple parts: pre-conditions generation submodule, pre-conditions checking submodule, pre-conditions analysis submodule, high-level plan adjustment submodule, and high-level plan reflection submodule.

Figure 1 is an example for a pre-conditions check module, containing simplified prompts and output examples of each submodule. The original current subtask is 'open microwave door', and the subtask after plan adjust is 'navigate to container'. Please check the Figure 1 for more details.

1. Pre-conditions generation submodule: use an LLM to generate pre-conditons for VLM to check the observation with. Return multiple string for checking.
2. Pre-conditions checking submodule: call a VLM to check whether pre-conditions are satisfied given side and wrist robot images. Return multiple boolean values to indicate the pre-conditions checking result.
3. Pre-conditions analysis submodule: call a LLM to extract whether the pre-conditions are satisfied. If not, analyze pre-conditions checking result and output the reason why they are not satisfied and recommended next steps.
4. High-level plan adjustment: given previous high-level plan and pre-conditions analysis result, adjust the original high-level plan, and instruct the robot to execute the first subtask (command).

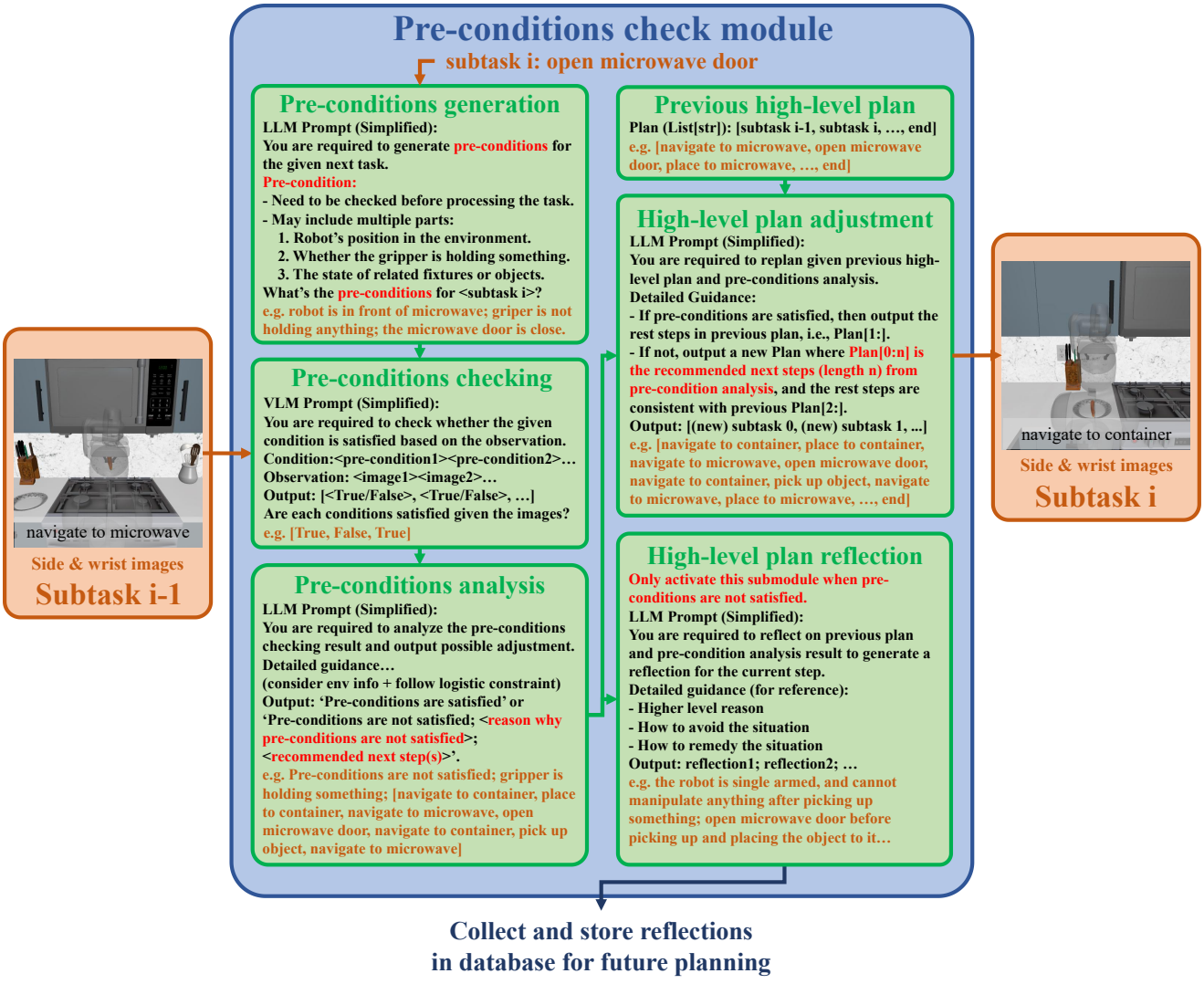

Figure 1: Simplified prompt example for pre-conditions check module, including pre-condition generation, checking, analysis, plan adjustment and reflection submodule. Please zoom in for detailed example. The post-condition check module is almost identical except for the missing reflection module.

5. High-level plan reflection: based on pre-conditions analysis result, further reflect on the ignorance in the environment and logical constraints in the initial high-level plan. The reflection are formalized into strings and will be stored in the database and used in the future.

As for the post-conditions check module, the only difference is the absence of the last high-level plan reflection submodule, as post-condition checking is meant to guarantee the successful execution of current subtask, which is not relevant to the high-level plan.

Discussion on Goal Description

This section will discuss how the description in goal prompt will affect the high-level plan generated by LLMs.

We tried to set 3 different types of goal prompt which has the same actual meaning, and conduct experiments on them using different LLMs:

1. Description 1: word order align with subtask order
2. Description 2: word order misalign with subtask order
3. Description 3: the simplest description

For example, in the experiment OpenMicrowavePnP, the 3 goal prompt are as follows (the microwave is initially closed, and is also stated in the scene description):

1. Description 1: open microwave door, pick up the vegetable from the counter and place it into the microwave
2. Description 2: pick up the vegetable from the counter and place it into the microwave, the microwave need to be opened first.

	gpt-4o	gpt-4o w/ reflection	DeepSeek-R1	DeepSeek-R1 w/ reflection
Description 1	65%	82.5%	72.5%	87.5%
Description 2	0%	17.5%	22.5%	77.5%
Description 3	0%	10%	17.5%	75%

Table 1: Initial Plan Success Rate for different models and settings. We use gpt-4o (Hurst et al. 2024) and DeepSeek-R1 (Guo et al. 2025) as representatives of models w/o and w/ reasoning ability. The result is the average of 10 trials each from 4 tasks. Description 1 - word order align with subtask order, Description 2 - word order misalign with subtask order, Description 3 - the simplest description.

3. Description 3: place the vegetable into the microwave.

We examine the initial plan outcome for different models, and compared the responses of models with and without reasoning capabilities. The results are shown in Table 1.

The results indicated that the planning of large language models is easily affected by the goal word order, which appears in models with and without reasoning ability. Moreover, the reasoning-capable models like DeepSeek-R1 are more likely to leverage information provided by the reflection context, while the gpt-4o model tend to ignore the reflection tips and stick to wrong answer. So we use a reasoning model (DeepSeek-R1) to reflect and generate initial plan, and use a non-reasoning model (gpt-4o) to serve as condition checkers and plan adjusters for time saving.

References

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Mandlekar, A.; Nasiriany, S.; Wen, B.; Akinola, I.; Narang, Y.; Fan, L.; Zhu, Y.; and Fox, D. 2023. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*.

Mandlekar, A.; Xu, D.; Wong, J.; Nasiriany, S.; Wang, C.; Kulkarni, R.; Fei-Fei, L.; Savarese, S.; Zhu, Y.; and Martín-Martín, R. 2021. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*.

Nasiriany, S.; Maddukuri, A.; Zhang, L.; Parikh, A.; Lo, A.; Joshi, A.; Mandlekar, A.; and Zhu, Y. 2024. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*.