Robotics Toolformer: Synergizing Robotics Reasoning and Acting with Mixed-Modal Tools

Anonymous EMNLP submission

Abstract

Recent advancements in large language models have shown their effectiveness in high-level robot planning. However, these models have limitations when handling longer and more complex instructions like “stack a pyramid” or “solve the Tower of Hanoi”. In this paper, we propose an innovative approach called Robotics Toolformer (RT), which aims to enhance the mixed-modal reasoning and long-term decision-making capabilities of language model-based robotics agents. RT introduces a multimodal reasoning paradigm for unified tool usage. It employs reasoning-acting traces with external mixed-modal tools to assist the agent in deducing, tracking, and updating action plans for embodied planning. By incorporating and expanding these tools, the agent can leverage a wider range of capabilities, enabling it to perform more complex tasks with only in-context learning from a single mixed-modal demonstration. When evaluating the Cliport benchmark (Shridhar et al., 2022; Zeng et al., 2020), our proposed method surpasses the performance of the imitation learning-based method, which extensively pre-trains on robotics manipulation tasks, as well as the code-as-policies (Liang et al., 2022) approach. Our approach demonstrates superior performance and efficiency by effectively handling complex instructions and achieving higher task success rates through learning from a one-shot mixed-modal demonstration.

1 Introduction

LLMs (Brown et al., 2020) have been utilized in high-level robot planning(Vemprala et al., 2023; Liang et al., 2022; Ahn et al., 2022), where they enable robots to understand and interpret natural language instructions provided by humans. By leveraging their language generation capabilities, LLMs can translate textual instructions into executable actions, enabling robots to perform complex tasks. While LLMs have proven effective in certain robot planning scenarios, their capabilities to handle longer and more complex instructions remain limited. Tasks that involve multi-step processes or intricate operations pose challenges for LLMs, as they struggle to generate accurate and coherent action plans from such instructions. For example, instructions like “stack a pyramid” or “solve the Tower of Hanoi” takes robot to perform multiple actions in a specific sequence, each dependent on the success of the preceding one. The complexity of such instructions often exceeds the capabilities of traditional LLM-based approaches, leading to suboptimal performance and limited practical applicability in real-world scenarios.

The previous works (Liang et al., 2022; Vemprala et al., 2023) of code generation chain of thought (Wei et al., 2022) carry out several pre-defined human-make prompts to derive the plans using their own internal representations to generate thoughts and is not grounded in the physical world, which limits its ability to reason reactively or update its knowledge. This can lead to issues like hallucination and error propagation over the reasoning process. One way to reduce hallucination and increase the alignment between real embodied environments is to introduce multi-modal reasoning. Huang et al. (2022); Zeng et al. (2022); Ahn et al. (2022) adopt a pipeline of firstly ground the object detected in the scene to text, use a language model to generate domain-specific plans, and then use an imitation learning-based controller to execute the plans. However, in these works, the visual-to-text model is solely utilized for object detection purposes and does not actively contribute to the disassembly of complex instructions into action primitives that can be executed by the LLM agent. The challenge of handling complex long-horizon instructions has not been effectively addressed in previous multi-modal LLM-based methods.

To solve the challenge of LLM-based agents struggling at handling complex long-horizon instructions, and inspired by ReAct (Yao et al., 2022) and Toolformer (Schick et al., 2023) and code-as-policies (Liang et al., 2022), we pro-

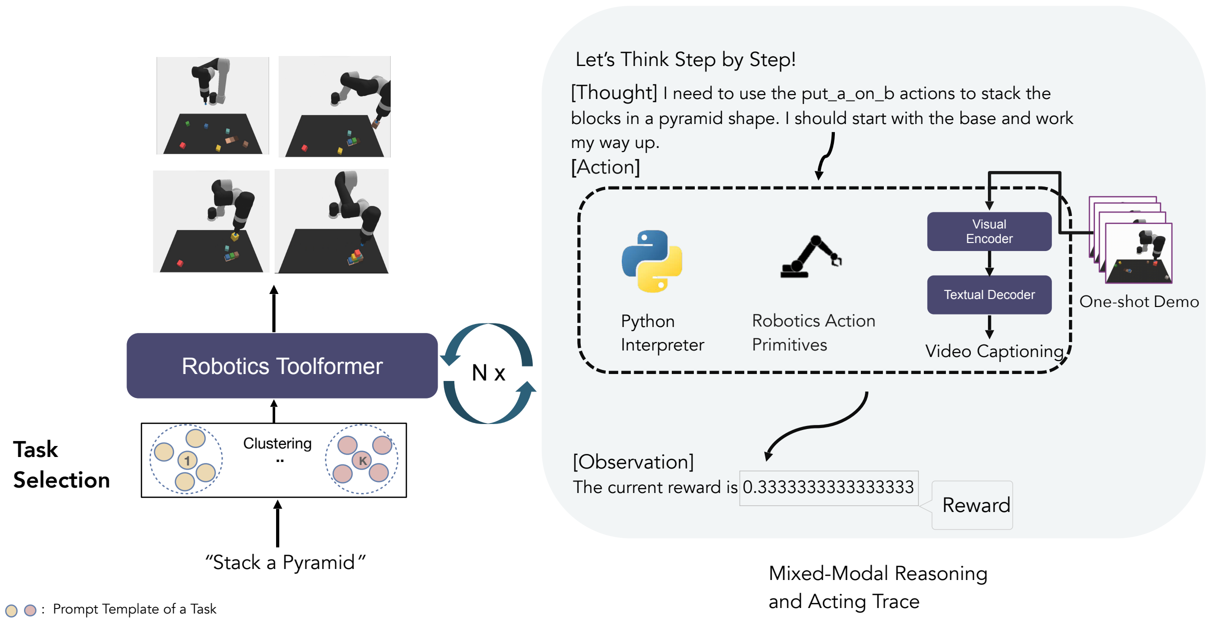

Let's Think Step by Step!
[Thought] I need to use the put_a_on_b actions to stack the blocks in a pyramid shape. I should start with the base and work my way up.
[Action]
Python Interpreter
Robotics Action Primitives
Visual Encoder
Textual Decoder
Video Captioning
One-shot Demo
[Observation]
The current reward is 0.3333333333333333
Reward
Robotics Toolformer
N x
Task Selection
Clustering
"Stack a Pyramid"
Prompt Template of a Task
Mixed-Modal Reasoning and Acting Trace

Figure 1: The Robotics Toolformer is an LLM-based agent that can use mixed-modal tools to solve complex long-horizon embodied tasks. The instruction is sent to the task selection modules for prompt template selection. The robotics toolformer then produces a mixed-modal reasoning-acting trace comprising thought: a natural language description of what should be done next, action: tool selection and action proposing, and observation: the feedback of the proposed action from the embodied environment.

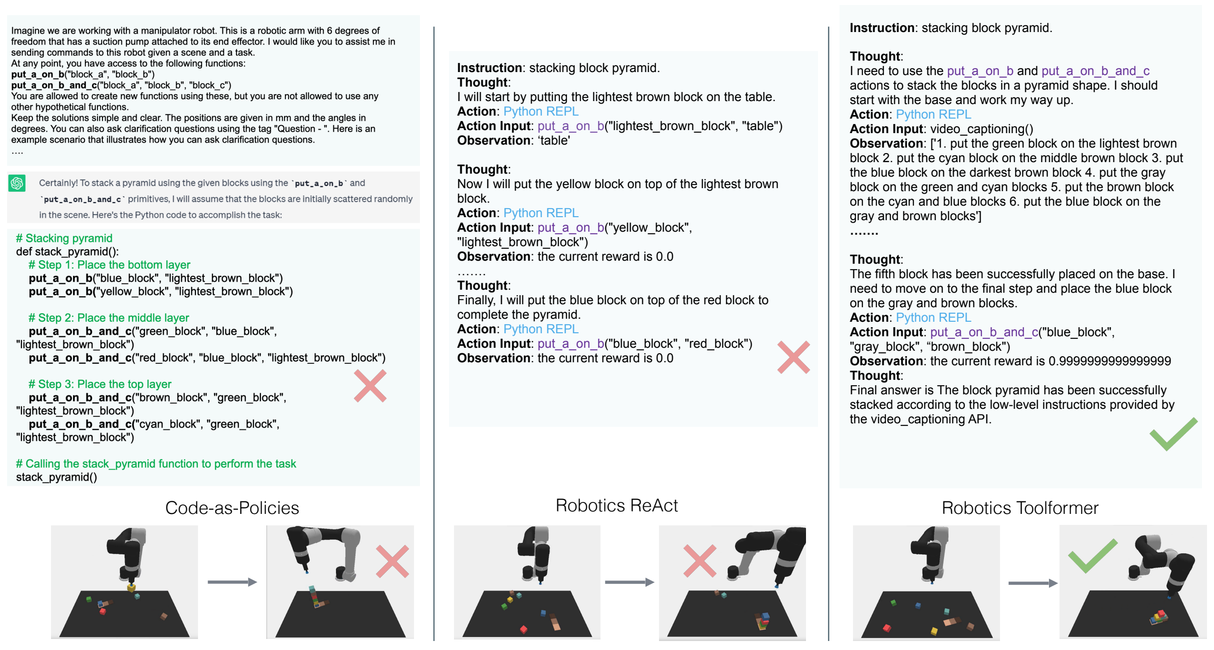
pose ***Robotics Toolformer***(RT), an embodied intelligence paradigm to combine mixed-modal reasoning and acting traces with toolformer for solving diverse robotics tasks. Robotics Toolformer prompts LLMs to generate both reasoning traces and actions pertaining to a task in an interleaved manner, which allows the model to perform dynamic reasoning to create, maintain, and adjust high-level plans for acting (reason to act), while also interacting with the external embodied environments by mixed-modal tool augmentation (e.g., python interpreter, multi-modalities pre-trained models, etc.) to incorporate additional information into reasoning.

To summarize, our key contributions are the following:

- ***Multimodal Reasoning for Unifed tool using:*** We introduce the Robotics Toolformer framework, which addresses the challenges faced by LLM-based agents when handling complex long-horizon instructions. This framework enables the unified use of tools from different modalities in embodied control, enhancing the agent's reasoning and acting capabilities.
- We demonstrate the superior performance of our approach on long-horizon tasks compare to previous LLM-based approaches. Furthermore, our results are comparable to Cliport with instruction oracle.

2 Related Works

Chain of Thought for Embodied Intelligence. LLMs exhibits impressive in-context learning capabilities for helping robotics planning (Ahn et al., 2022; Zeng et al., 2022; Vemprala et al., 2023). The prompting method that is mostly taken in robotics controlling is Chain-of-Thought(Cot) (Wei et al., 2022; Driess et al., 2023; Liang et al., 2022). However, according to (Yao et al., 2022), Cot reasoning is a static black box, in that the model uses its own internal representations to generate thoughts and is not grounded in the external world, which limits its ability to reason reactively or update its knowledge. Vanilla Cot can lead to issues like fact hallucination and error propagation over the reasoning process. One attempt to mitigate the problem of hallucination is multi-modal chain-of-Thought (Zhang et al., 2023), which incorporates language (text) and vision (images) modalities into a two-stage framework that separates rationale generation and answer inference. Another is ReAct (Yao et al., 2022), which generates both reasoning traces and



Imagine we are working with a manipulator robot. This is a robotic arm with 6 degrees of freedom that has a suction pump attached to its end effector. I would like you to assist me in sending commands to this robot given a scene and a task.
At any point, you have access to the following functions:
put_a_on_b("block_a", "block_b")
put_a_on_b_and_c("block_a", "block_b", "block_c")
You are allowed to create new functions using these, but you are not allowed to use any other hypothetical functions.
Keep the solutions simple and clear. The positions are given in mm and the angles in degrees. You can also ask clarification questions using the tag "Question - ". Here is an example scenario that illustrates how you can ask clarification questions.
....

Certainly! To stack a pyramid using the given blocks using the `put_a_on_b` and `put_a_on_b_and_c` primitives, I will assume that the blocks are initially scattered randomly in the scene. Here's the Python code to accomplish the task:

```
# Stacking pyramid
def stack_pyramid():
   # Step 1: Place the bottom layer
   put_a_on_b("blue_block", "lightest_brown_block")
   put_a_on_b("yellow_block", "lightest_brown_block")

   # Step 2: Place the middle layer
   put_a_on_b_and_c("green_block", "blue_block",
"lightest_brown_block")
   put_a_on_b_and_c("red_block", "blue_block", "lightest_brown_block")

   # Step 3: Place the top layer
   put_a_on_b_and_c("brown_block", "green_block",
"lightest_brown_block")
   put_a_on_b_and_c("cyan_block", "green_block",
"lightest_brown_block")

# Calling the stack_pyramid function to perform the task
stack_pyramid()
```

Instruction: stacking block pyramid.
Thought:
I will start by putting the lightest brown block on the table.
Action: Python REPL
Action Input: put_a_on_b("lightest_brown_block", "table")
Observation: 'table'

Thought:
Now I will put the yellow block on top of the lightest brown block.
Action: Python REPL
Action Input: put_a_on_b("yellow_block", "lightest_brown_block")
Observation: the current reward is 0.0
....
Thought:
Finally, I will put the blue block on top of the red block to complete the pyramid.
Action: Python REPL
Action Input: put_a_on_b("blue_block", "red_block")
Observation: the current reward is 0.0

Instruction: stacking block pyramid.

Thought:
I need to use the put_a_on_b and put_a_on_b_and_c actions to stack the blocks in a pyramid shape. I should start with the base and work my way up.
Action: Python REPL
Action Input: video_captioning()
Observation: ['1. put the green block on the lightest brown block 2. put the cyan block on the middle brown block 3. put the blue block on the darkest brown block 4. put the gray block on the green and cyan blocks 5. put the brown block on the cyan and blue blocks 6. put the blue block on the gray and brown blocks']
....

Thought:
The fifth block has been successfully placed on the base. I need to move on to the final step and place the blue block on the gray and brown blocks.
Action: Python REPL
Action Input: put_a_on_b_and_c("blue_block", "gray_block", "brown_block")
Observation: the current reward is 0.9999999999999999
Thought:
Final answer is The block pyramid has been successfully stacked according to the low-level instructions provided by the video_captioning API.

Figure 2: Comparison of 3 prompting methods, from left to right: Code as Policies, Robotics ReAct, and Robotics Toolformer on solving the Stack-A-Pyramid task in Cliport benchmark. Only the RT can successfully execute the complex long-horizon task without being informed with object information in the sense.

task-specific actions in an interleaved manner. Toolformer (Schick et al., 2023) fine-tuned the LLMs with the dataset containing APIs to make LLMs able to decide for themselves when and how to use which tool. Inspired by previous works, we propose ***Robotics Toolformer***, which is a code generation LLM-based agent that generates mixed-modal reasoning and acting traces with mixed-modal tool augmentation for long-horizon robotics control.

Most Similar Works. The work that is mostly similar to ours is Mind's Eye (Liu et al., 2022). To facilitate physical reasoning, the agent of (Liu et al., 2022) generates a video simulation of certain physical evolution, uses sensors to record the sensory physical quantities, then reasoning via Cot. We extend the mind's eye (Liu et al., 2022) to embodied planning by replacing Cot with mixed-modal ReAct in Python code and tools augmentation. We list the difference between our method with the existing methods in Table 1.

Method	Mixed-Modal	Tool	Cot Type
Code-as-Policies	×	Py	Code Generation Chain-of-Thought
ReAct	×	TITO	Reason-Act Traces
Toolformer	×	TITO	Cot with Tools
Mind's Eye	✓	Physical, PI	Cot with Simulator
Robotics Toolformer	✓	Py, V2T	Mixed-Modal ReAct

Table 1: Comparison with the existing methods. Py,V2T, TITO, Physical represents Python Interpreter, Visual-to-Text Model, Text-in-text-out APIs, Physical Engine respectively.

3 Robotics Toolformer

3.1 Problem Formulation

Consider a general setup where a robotics agent interacts with an embodied environment to solve tasks. At the time t, the agent receives an instruction $\mathcal{I}$ and samples an observation $o_t \in \mathcal{O}$ from the embodied environment, as defined in (Zeng et al., 2020). The objective of the LLMs is to determine the intermediate sub-instruction or code snippet, denoted as action input i_t, and the corresponding action a_t to execute. Following the execution of the action, the agent can sample a reward signal r_t from the environment. In contrast to the original ReAct framework (Yao et al., 2022), which solely considers simple tools from the Wikipedia API, our proposed framework, ***Robotics Toolformer***, extends the tool space $\mathcal{A}$ to include mixed-modal tools, such as a visual-to-text model, a Python interpreter and more. By incorporating this expanded tool space, the agent can leverage diverse external tools to enhance its mixed-modal reasoning and acting capabilities. To detect the success state, we incorporate a reward function as a monitoring tool, determining whether an action is successfully proposed and if the agent is ready to propose the next action. We define the trajectory of the RT as a sequence of observations, actions, action inputs, and rewards at each time step t. Specifically, the trajectory at time t is denoted as $c_t = (o_1, i_1, a_1, r_1, ..., o_t, a_t, i_t, r_t)$.

Our approach expands the action space of LLMs, denoted as $\mathcal{L}$, by integrating external tools from the tool space $\mathcal{A}$. In our work, the action input i_t and the reasoning trace (thought) are generated by the LLMs without directly affecting the external environment, resulting in no observation feedback. Instead, the purpose of the thought is to compose useful information by reasoning over the current trajectory c_t and updating the trajectories to support future reasoning and acting. Furthermore, during each iteration, the LLM must determine which tool to utilize from the tool space $\mathcal{A}$ based on the trajectories. We refer to the process of determining thought, action input, and action with mixed-modal tool augmentation as a mixed-modal Reasoning-acting trace (mixed-modal ReAct). To ensure the LLM agent avoids hallucination, we directly execute the tools in the mixed-modal ReAct to ground the language plans into real-world interactions. We report the results of these interactions back to the chain as observations o_t, which guide further reasoning and acting. The objective of the agent is to fulfill the instruction I by achieving a fully marked reward of 1.0.

3.2 Mixed-Modal Action Space

we adopt tools to enhance the agent's reasoning and acting capabilities in different aspects. Firstly, we utilize the Python REPL API, which serves as a Python shell enabling the execution of Python code snippets. This tool allows the agent to interact with the Python programming language, enabling it to leverage the rich ecosystem of libraries and perform the actions primitives defined in Cliport benchmarks (Shridhar et al., 2022; Zeng et al., 2020). Moreover, we leverage a video-to-text pretrained model as the module to propose multimodal reasoning. This multimodal tool differs from previous approaches used in tabletop robotics grasping methods (Zeng et al., 2022; Huang et al., 2022), which relied solely on image-to-text models for simple object detection. Utilizing a video-to-text model, our goal is to streamline the process of interpreting complex instructions and converting them into actionable steps in natural language. This approach combines visual information with textual instructions, giving the agent a better understanding of the task and facilitating more comprehensive reasoning. Consequently, given high-level instruction e.g. "Solve the Tower of Hanoi" and visual demonstrations, our agent can deduce specific intermediate actions such as "1. Move the red ring to the lighter brown side. 2. Move the green ring to the darker brown side. ...".

Additionally, we incorporate a cumulative reward function from Cliport (Zeng et al., 2020). The reward function serves as a measure of the agent's performance and provides feedback directly from the embodied environment. Specifically, the reward function assesses the agent's actions based on how well they align with the desired objectives of the task. It provides positive rewards for actions that contribute to task completion and zero rewards for actions that deviate from the desired outcomes or violate safety constraints. The working mechanism of the cumulative reward function is dependent on a human-designed task-specific oracle. Before the agent proposes actions, the oracle pack all the subgoals in a list, as well as the success metrics. If the metric is 'pose', it calculates the reward based on object poses and matches with target poses. For each object, it checks if its pose matches any of the targets' poses and assigns a portion of the maximum reward to the step reward variable. If the metric is 'zone', it calculates the reward based on the intersection of objects with zones. It iterates over each zone and counts the number of valid points (object points within the zone). The step reward is determined by the ratio of valid points to total points, multiplied by the maximum reward. If the current goal step is complete (i.e., the step reward is close to the maximum reward), the progress is updated, and the current goal step is removed from the goals list. Finally, the reward will be returned in the observation section of ReAct (Yao et al., 2022) chains.

3.3 Translate Highly Complex Visual Manipulation to Feasible Actions

Network Architecture. In our work, we employ a network architecture specifically designed for video captioning tasks. Our chosen network structure is the Generative Image-to-text Transformer, as introduced in the work by (Wang et al., 2022). This architecture serves as a tool for translating highly complex visual robotics manipulations into feasible pre-defined action primitives with textual

Algorithm 1: Robotics Toolformer Pseudocode with Mixed-Modalities Reasoning

```
Input: Textual Instruction $\mathcal{I}$ and visual Demonstration $\mathcal{D}$
Output: trajectories of embodied interactions
Initialize trajectory $c$ = [];
while task not completed do
    1. Select the suitest prompt template via Bert
    2. Observe current environment state $o_t$;
    3. Append $o_t$ to trajectory $c$;
    4. Generate action input $i_t$ using LLMs and reasoning over $c$;
    5. Select tool $a_t$ from $\mathcal{A}$ based on $i_t$ and $c$;
    6. Execute $a_t$ in the environment;
    7. Obtain reward $r_t$ from the environment;
    8. Append $i_t$, $a_t$, and $r_t$ to trajectory $c$;
    if success state detected then
        return the final trajectory $c$;
    end
end
```

descriptions. The visual encoder component of the network is built upon the contrastive pre-trained model Florence (Yuan et al., 2021). It takes raw video frames as input and produces a compact 2D feature map. This feature map is then flattened into a list of individual features. To further process these features, an additional linear layer and a layer normalization layer are applied. This projection step transforms the image features into a D-dimensional space, which serves as the input for the text decoder. we sample multiple frames from each rendering sequence of Cliport (Zeng et al., 2020) dataset and encode each frame via the image encoder independently. Afterward, we add a learnable temporal embedding (initialized as zeros), and concatenate the features from sampled frames. The text decoder, implemented as a GPT-3 architecture (Brown et al., 2020), predicts a sequence of feasible robotics actions in language. The transformer module comprises multiple transformer blocks, each consisting of a self-attention layer and a feed-forward layer. To prepare the text input, it is tokenized and embedded into D dimensions. Positional encoding is added to account for the sequential order of the tokens, followed by a layer normalization step. In the robotics video captioning process, the video frames' features are concatenated with the text embeddings to form the input for the transformer module. The decoding of the text begins with the special [BOS] (beginning of the sentence) token, and subsequently, the transformer generates the output text in an auto-regressive manner until it encounters the [EOS] (end of the sentence) token or reaches the maximum number of decoding steps.

Fine-tuning for Robotic Video Captioning Model. To fine-tune the GIT model for robotic video captioning, we apply the language modeling (LM) loss. For each video-text pair, denoted as $v_i \in \mathcal{V}$ for the video frames and y_i for the text tokens, we use cross-entropy loss with label smoothing of 0.1. The loss is computed as the average cross-entropy over all tokens in the sequence, including the [BOS] and [EOS] tokens.

$$l = \frac{1}{N+1} \sum_{i=1}^{N+1} \mathrm{CE}\left(y_i, p\left(y_i \mid \mathcal{V}, \{y_j, j = 0, \cdots, i-1\}\right)\right) \tag{1}$$

This loss function encourages the model to accurately predict the next token in the sequence based on the video frames and previously generated tokens. It is important to note that our fine-tuning process allows each iteration to predict all tokens, making it more efficient for training large-scale robotic video captioning models. It is worth mentioning that our approach differs from the original GIT model, which is primarily designed for high-level one-sentence summarization of videos. However, in our work, we leverage the GIT architecture and adapt it specifically for robotic video captioning, enabling more detailed and action-oriented descriptions of the visual robotics manipulations in the videos.

3.4 Mixed Modalities ReAct Chains

According to the Code-as-policies (Liang et al., 2022), LLM-based agents face challenges when interpreting longer and more complex commands, such as "build a house with the blocks." This difficulty arises due to the limitations of LLMs in understanding the scope of defined perception APIs. Furthermore, existing visual-language models lack the capability to describe qualitative aspects, like whether a trajectory is "bumpy" or "more C-shaped." Another issue when dealing with complex and long-horizon tasks is the increased risk of hallucination. As the planning horizon extends, it becomes challenging to monitor the robot's actions and accurately detect the success state. To address these challenges and enable LLM-based agents to manipulate complex, long-horizon tasks, we draw inspiration from the work of multi-modal chain-of-thought (Zhang et al., 2023). By incorporating multi-modal reasoning into the ReAct chain of thought, we can overcome these limitations. We introduce visual demonstrations into the mixed-modal ReAct by converting paired video frames into action sequences. This process helps the agent understand how to parse highly complex high-level instructions into feasible action primitives. By leveraging both visual and linguistic information, the agent gains a better understanding of the task requirements and can generate more effective action plans. To further mitigate the challenges, we also propose using reward functions that provide feedback on the proximity of the current state to the final state. This allows the agent to track its progress and make informed decisions during the execution of long-term actions. By incorporating mixed modalities and leveraging visual demonstrations, multi-modal reasoning, and informative reward functions, we enhance the capabilities of LLM-based agents in handling complex, long-horizon tasks. This approach facilitates better interpretation of complex commands, mitigates the risk of hallucination, and enables effective monitoring and progress tracking throughout the task execution.

4 Experiments

We conduct empirical evaluations of RT against state-of-the-art baselines, including Cliport (Shridhar et al., 2022; Zeng et al., 2020) and LLM-based methods (Liang et al., 2022) on the benchmark proposed in Cliport (Zeng et al., 2020; Shridhar et al., 2022). Our primary focus is to evaluate the agents' ability to solve long-horizon tasks with abstract instructions. Notably, we employ the same action primitives as defined in Cliport (Shridhar et al., 2022) and Code-as-Policies (Liang et al., 2022). We discover that incorporating mixed-modal tools significantly alleviates the problem of hallucination. By grounding the one-shot visual demonstration prompts and high-level instructions for complex robotics manipulations (e.g., "Stacking a pyramid") to combinations of intermediate instructions containing action primitives in natural language, we effectively address the issue. Our evaluation encompasses five long-horizon tasks from the Cliport benchmark, and the results demonstrate that our method outperforms previous code-as-policies approaches. Our method achieves comparable performance to Cliport (Shridhar et al., 2022), which is prompted by ground-truth intermediate actions in the language (instruction oracle), instead of abstract high-level instruction.

4.1 Setup

Dataset. We adopt the language-conditioned manipulation tasks in Cliport (Shridhar et al., 2022) as evaluation datasets. Each task instance is constructed by sampling a set of objects and attributes: poses, colors, sizes, and object categories.
As Code-as-Policies (Liang et al., 2022) has already proven its effectiveness in short-horizon tasks, In our evaluation, we evaluate five tasks proposed in Cliport: Stack Pyramid, Towers of Hanoi, Assembling Kits, Put Block in Bowl, Packing shapes. The first three tasks are long horizon tasks with abstract instructions, which means that we only prompt the agent with very brief instructions involving complex concepts, i.e. "Stack a Pyramid", "Solve the Tower of Hanoi", and "Assembling all kits to the corresponding position", respectively. Put Block in Bowl is a long-horizon combinatorial task that contains combinations of sub-instructions i.e. "1. put the blue block in the blue bowl. 2. put the red block in the red bowl ..." Packing shape is a short horizon task that directly prompts the agent to pick what object and place it where.

Simulation Environment. All simulated experiments are based on a Universal Robot UR5e with a suction gripper. The setup provides a systematic and reproducible environment for evaluation, especially for benchmarking the ability to ground semantic concepts like colors and object categories. The input observation is a top-down RGB-D reconstruction from 3 cameras positioned around a rectangular table: one in the front, one on the left shoulder, and one on the right shoulder, all pointing towards the center. Each camera has a resolution of 640×480 and is noiseless.

Evaluation Metric. Same as Cliport (Shridhar et al., 2022), we adopt the 0 (fail) to 100 (success) scores proposed in the Ravens (Zeng et al., 2020) benchmark. The score assigns partial credit based on the task, e.g., 5/6 for fulfilling 5 sub-instruction in stacking pyramid tasks. See Appendix A.1 for detailed metrics information. During an evaluation

Method	**Packing Shape**	**Put Blocks in Bowl**	**Stack Block Pyramid**	**Tower of Hanoi**	**Assembling Kits**
Code-as-Policies	100.0	97.0	0.0	0.0	0.0
Cliport(1-shot)	29.0	43.0	28.4	61.6	9.6
Cliport(10-shot)	42.0	69.5	51.1	86.6	28.2
Cliport(100-shot)	47.0	77.4	64.3	89.1	37.6
Cliport(1000-shot)	41.0	72.6	59.4	83.5	38.9
Robotics ReAct	93.0	99.0	0.0	0.0	0.0
Robotics Toolfromer	75.0	95.0	**80.0**	**99.0**	**99.0**

Table 2: Comparison of Robotics Toolformer with Code-as-Policies, Cliport, and Robotics. When it comes to long-horizon tasks involving complex concepts in the instruction for the last three tasks, both Code-as-polices and Robotics ReAct show extremely poor performance. RT gets superior performance on these than the baselines.

episode, the agent keeps interacting with the scene until an oracle indicates task completion.

Baselines. To study the effectiveness of the RT We mainly evaluate our proposed method against the following existing approaches on the Cliport (Shridhar et al., 2022) benchmark:

- **Imitation-learning based method with instruction oracle.** For each task in 4.1, we report the result of Cliport (Shridhar et al., 2022) model that is pre-trained with 1, 10, 100, and 1,000 demonstrations. An oracle is employed to prompt the agent with ground truth intermediate actions in language.
- **Code as Policies based on GPT-3.5** (Liang et al., 2022; Vemprala et al., 2023), a code generation method that adopt chain-of-thought for solving embodied tasks.
- **Robotics ReAct** (Yao et al., 2022), we also compare with the only-language code generation Reasoning-Acting traces for the purpose of ablation analysis. In this ablation experiment, we investigate how the mixed-modal reasoning module enhances robotics planning.

For all LLM-based methods, we adopt the latest OpenAI language model GPT-3.5-turbo-0613 (Ouyang et al., 2022) as the planner.

4.2 Main Result.

Table 2 presents results from our experiments in long-horizon robotics manipulation tasks. These results support our main claims that ReAct with mixed-modal tool augmentation can significantly enhance the performance of solving long-horizon embodied tasks. We observe that both Code-as-Policies (Liang et al., 2022) and ReAct (Yao et al., 2022) perform worse than the imitation learning-based method(Cliport) on the Stack Pyramid, Towers of Hanoi, and Assembling Kit tasks, failing to solve any instances. This poor performance can be attributed to the difficulty in understanding the concepts of "Hanoi tower" and "pyramid" using language-only internal representations. Instead, these concepts need visual knowledge to understand. Figure 2 compares the different LLM-based methods in solving the Stack-Block-Pyramid task. Notably, the augmentation of mixed-modal tools enables RT to achieve success rates of 80.0%, 99.0%, and 99.0% on the Stack-Block-Pyramid, Tower-of-Hanoi, and Assembling-Kits tasks, respectively. However, LLM-based agents show superior performance in detecting objects with objective attributes such as color and position compared to IL-based agents and RT. It is important to note that code-as-polices (Liang et al., 2022) simplifies the gathering of object information by utilizing ground-truth primitives, but ours does not have access to the ground-truth information. The "Packing-Shape" task provides specific details on which objects to place in the box, eliminating the necessity to decipher high-level concepts into low-level instructions. This direct guidance for picking and placing enables code-as-policies to effectively handle any given instruction. However, our method uses multi-modal reasoning to infer which object to pick and where to place it. Furthermore, multi-task training of GIT may bring bias to long-horizon tasks. The above reasons might lead to RT's lower performance on "Packing-Shape", compared to code-as-policies. Overall, The results from table 2 highlight one key challenge that LLM-based embodied agents face: understanding visual concepts through language grounding and acquiring the ability to accomplish goals that involve complex

	GIT		
Training Steps	500	1,000	3,000
BeLU	0.41	0.74	0.96
ROUGE	0.68	0.82	0.98
METEOR	0.70	0.82	0.98

Table 3: The evaluation of GIT's performance, when training the model for different steps: 500, 1000, and 3,000. The metrics are computed by taking the average across all tasks.

Agent	Type	Stack-Block-Pyramid	Assembling-Kits
Robotics ReAct	zero-shot	0.0	0.0
	Oracle	96.0	100.0
Robotics Toolformer	500	0.0	0.0
	1,000	11.0	0.0
	3,000	80.0	99.0

Table 4: The assessment of two different types of Robotics ReAct and RT models with various GIT, considering varying numbers of training steps.

visual concepts using practical action primitives.

4.3 Ablation Study

In our study on ablation research, we primarily examine the impact of mixed-modal video-to-text models on the planning accuracy of RT.

Setup. We explore the performance of the following LLM-based agent on the selected long-horizon tasks (Stack-Block-Pyramid and Assembling Kits): 1. RT utilizing GIT (Wang et al., 2022) for various training durations: 500, 1,000, and 3,000 training steps. 2. Robotics ReAct agent performing without any prior object information (zero-shot). 3. Robotics ReAct agent guided by an Instruction Oracle.

Mixed-modal Reasoning Model Finetuning. We implement our video-to-text model for robotics task captioning by building upon the GIT large (Wang et al., 2022), it leverages the pre-trained CLIP/ViT-L/14 architecture (Radford et al., 2021) as the visual encoder. For detailed information regarding the network architectures, hyper-parameters, and training specifics, please refer to section A.2 in the paper. Our implementation of GIT (Wang et al., 2022) are trained from 400$\mathbb{T}$ demonstration of the dataset in Section 4.1 and evaluate on 100$\mathbb{T}$ corresponding validation dataset, note that $\mathbb{T}$ denotes the number of tasks in Section 4.1. The metrics we use to evaluate the video captioning model are BLEU (Papineni et al., 2002), Rouge (Lin, 2004), and METEOR (Banerjee and Lavie, 2005). In Table 3, we report the multi-task performance by taking the average score of all tasks on certain metric.

Ablation Results. The ablation study presented in Table 4 examines the different variations of GIT. The results indicate that even though the GIT (1000 steps) achieves decent performance with BELU 0.74, ROUGE 0.82, and METEOR 0.82, it is still inadequate for handling long-horizon tasks effectively. However, when GIT (3000 steps) is employed with improved metrics such as BELU 0.96, ROUGE 0.98, and METEOR 0.98, the RT can achieve 80.0 on Stack-Block-Pyramid and 99.0 on Assembing-Kits. This result indicates the RT requires the video-to-text model to be very accurate and precise in order to effectively execute embodied planning.

5 Conclusion

The Robotics toolformer is a multimodal reasoning paradigm for unified tool using. Inserting mixed-modal tools into the reasoning-acting can reduce hallucination and enhance the alignment between the inner representation of LLM with the embodied environment. Our experiments have demonstrated that our method outperforms current approaches in tackling intricate long-term robotics planning tasks using just one demonstration with mixed modalities. This framework has promising potential for even broader applications where other pre-existing powerful tools could be integrated to solve more embodied tasks.

6 Limitations

Although RT provides a viable approach for enabling LLM-based agents to solve complex tasks with long-term goals, there are several limitations that need to be addressed. One limitation is the lack of consideration for mixed-modal tools in learning policies to connect complex instructions with feasible primitives at scale. This is due to the limited availability of long-horizon visual-to-language datasets in embodied intelligence. Additionally, our work still relies on one-shot demonstrations for in-context learning and transforming abstract concepts into primitives. Future research could explore the approach to achieve zero-shot mixed-modal reasoning via tool augmentation.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. 2022. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2022. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR.

Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res*, 2:20.

Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. 2022. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. 2021. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*.

Andy Zeng, Peter R. Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. 2020. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv: Robotics*.

Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

A Appendix

A.1 Cliport Dataset and Evaluation Metrics

As for the benchmark for evaluating our proposed method with the baselines, we use the following datasets from Cliport (Shridhar et al., 2022). Note that when we use LLM-agent to execute the instructions, we do not give step-by-step ground-truth intermediate instructions to the agent like Cliport, instead, we ask the agents themselves to infer the intermediate instructions.

1. Assembling Kits **Task:** Precisely place each specified shape in the specified hole following the order prescribed in the language instruction generated by the LLM-based agent at each timestep. This is one of the hardest tasks in the benchmark requiring precise placements of shapes of randomized colors and grounding spatial relationships. Each task instance contains 5 shapes and a kit with randomized poses. **Goal:** assembling all the kits to the corresponding holes. **Success Metric:** The pose of each shape matches the specified hole at the correct timestep. The final score is the total number of shapes that were placed in the correct pose at the correct timestep, divided by the total number of shapes in the scene (always 5).

2. Stack Block Pyramid. **Task:** Build a pyramid of colored blocks in a color sequence specified through the step-by-step language instructions generated by the LLM-based agent. Each task contains 6 blocks with randomized colors and 1 rectangular base, all initially placed at random poses. **Goal:** stack a pyramid by 6 blocks and 1 rectangular base. **Success Metric:** The pose of each block at the corresponding timestep matches the specified location. The final score is the total number of blocks in the correct pose at the correct timestep, divided by the total number of blocks (always 6).

3. Towers of Hanoi. **Task:** Move the ring to the specified peg in the LLM-based agent-generated language instruction at each timestep. The sequence of ring placements is always the same, i.e. the perfect solution to three-ring Towers of Hanoi. This task can be solved without using colors by just observing the ring sizes. However, it tests the agent's ability to ignore irrelevant concepts to the task (color in this case). The task involves precise pick and place actions for moving the rings from peg to peg. **Goal:** Solve the tower of Hanoi via moving 3 rings (small, medium, and big) across 1 peg base. **Success Metric:** The pose of each ring at the corresponding timestep matches the specified peg location. The final score is the total number of correct ring placements, divided by the total steps in the perfect solution (7 for three-ring Towers of Hanoi).

4. Put Blocks in Bowl. **Task:** Place all blocks of a specified color in a bowl of the specified color. Each bowl fits just one block and all scenes contain enough bowls to achieve the goal. Each task instance contains several distractor blocks and bowls with randomized colors. The solutions to this task are multi-modal in that there could be several ways to place the blocks specified in the language goal. This task does not require precise placements and mostly tests an agent's ability to ground color attributes. **Goal:** to place blocks with certain colors on the bowls with certain colors **Success Metric:** All blocks of the specified color are within the bounds of a bowl of the specified color. The final score is the total number of correct blocks in the correct bowls, divided by the total number of relevant color blocks in the scene.

Packing Shapes **Task:** Place a specified shape in the brown box. Each task instance contains 1 shape to be picked along with 4 distractor shapes. The shape colors are randomized but have no relevance to the task. This task does not require precise placements and is mostly a test of the agent's semantic understanding of arbitrary shapes. **Goal:** To pack the required shapes to the brown box **Success Metric**: The correct shape is inside the bounds of the brown box.

A.2 Details of Fine-tuning GIT

Data Processing. We generate the video dataset of Cliport (Shridhar et al., 2022) by extracting the RGB information with the corresponding intermediate instructions of each frame. The processed image size is 224 $\times$ 224.

Hyperparameters. The hyperparameter of our reproduction of GIT (Wang et al., 2022) are list at the Table 5

Hyper-param	**GIT**
parameters num	347M
learning rate	3e-5
training batch size	1
gradient accumulation step	32
max train steps	5000
mixed precision	fp16
num GPUs	1
GPU type	A100

Table 5: The hyperparameter of our reproduction of GIT, other hyperparameters not mentioned on the table are kept the same as original GIT