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

From LLMs to Actions: Latent Codes as Bridges in Hierarchical Robot Control

Yide Shentu*,† Philipp Wu*†,‡ Aravind Rajeswaran†,‡ Pieter Abbeel†
*Equal contribution
†University of California, Berkeley ‡Meta

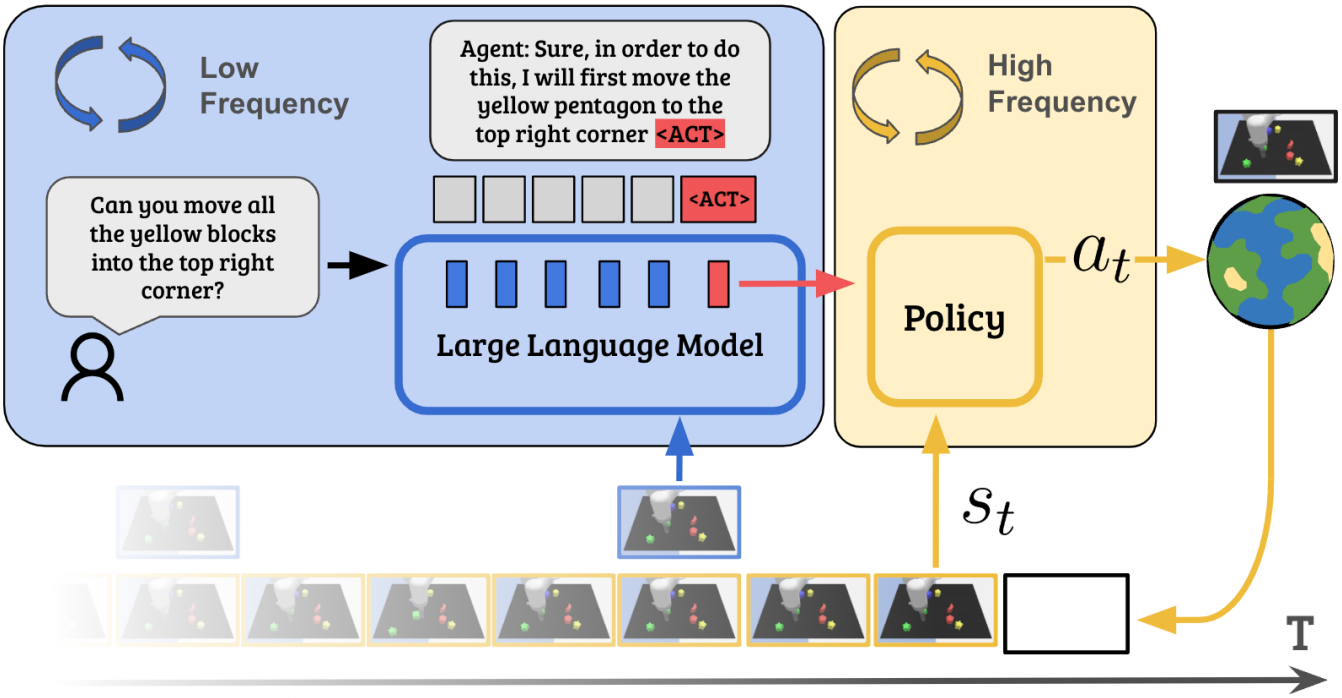


Fig. 1: Illustration of our proposed Latent Code as Bridges architecture. Given a high-level task description and the observation, a Large Language Model (LLM) generates a textual description of an action and an <ACT> token. The feature embedding from the <ACT> token's last layer serves as a high-level latent goal for the downstream policy network. Our modular hierarchical approach synergies the LLM's high-level reasoning with the pre-trained policy's responsive low-level control, addressing the limitations of direct low-level action output by monolithic LLMs. Unlike methods that using a LLM to directly output agent actions [1], our approach can run the LLM reasoning and action policy execution loops asynchronously, mirroring human-like task execution with immediate low-level feedback when interacting with the physical world and slower, deliberate reasoning when considering longer term planning. At test time, the action policy frequently updates actions based on environment changes and the latest <ACT> token's embedding, while the LLM updates are less frequent, enabling efficient, real-world inference.

***Abstract*— Hierarchical control for robotics has long been plagued by the need to have a well defined interface layer to communicate between high-level task planners and low-level policies. With the advent of LLMs, language has been emerging as a prospective interface layer. However, this has several limitations. Not all tasks can be decomposed into steps that are easily expressible in natural language (e.g. performing a dance routine). Further, it makes end-to-end finetuning on embodied data challenging due to domain shift and catastrophic forgetting. We introduce our method – Latent Codes as Bridges (LCB) – as an alternate architecture to overcome these limitations. LCB uses a learnable latent code to act as a bridge between LLMs and low-level policies. This enables LLMs to flexibly communicate goals in the task plan without being entirely constrained by language limitations. Additionally, it enables end-to-end finetuning without destroying the embedding space of word tokens learned during pre-training. Through experiments on Language Table and Calvin, two common language based benchmarks for embodied agents, we find that LCB outperforms baselines (including those w/ GPT-4V) that leverage pure language as the interface layer on tasks that require reasoning and multi-step behaviors.**

I. INTRODUCTION

The field of robotics has long oscillated between two predominant architectural paradigms for enabling agents to solve complex tasks. At one end of the spectrum, we have seen **modular hierarchical policies** [2], [3] for control that leverage rigid layers like symbolic planning, trajectory generation, and perception. On the other end are **end-to-end policies** [4], [5] that directly map sensory observations to actions through high-capacity neural networks. This dynamic history reflects the ongoing quest to reconcile the logical

human-like reasoning with the flexible dexterity of human motor control.

The advent of *large language models* (LLMs) [6], [7] and their remarkable language interpretation and reasoning capabilities have reignited interest in hierarchical control architectures. Recent works [8], [9], [10] have leveraged LLMs and *Multimodal Large Language Models* (abbreviated as LLMs in this paper unless specified otherwise) in place of high-level symbolic planners, enabling impressive results like mobile rearrangement of objects based on open-vocabulary instructions [11]. Despite these advances, the core deficiencies of hierarchical architectures remain – namely the need for a set of clearly defined control primitives and an interface between layers in the hierarchy. For example, LLMs leverage the semantic meaning of action verbs to coordinate low-level primitives like *go-to*, *pick*, *place* etc. However, we humans perform a variety of movements with our body that contribute to our dexterity and daily function, yet **cannot be easily described using language.**

In this backdrop, we present **L**atent **C**odes as **B**ridges, or LCB, a new policy architecture for control that combines the benefits of modular hierarchical architectures with end-to-end learning (see Fig. 1 for an illustration). Specifically, LCB can not only directly leverage LLMs for high-level reasoning and pre-trained skills/policies for low-level control, but also improve these components with end-to-end learning to transcend their initial capabilities. This is achieved by learning an <ACT> token at the interface layer which can modulate the low-level policies. As a result of this choice, LCB can overcome the inherent limitations of solely relying on language as the interface layer, since several behaviors are hard to describe in language. Additionally, by leveraging a separate <ACT> token, we do not erase the core language generation and reasoning capabilities of the LLM during finetuning. We test LCB on a series of long-horizon and reasoning tasks in Language Table [12] and Calvin [13], two common language based benchmarks for embodied agents. We find that LCB considerably outperforms baselines that leverage LLMs to sequence low-level skills using pure language as the interface layer. See our website for more.

II. RELATED WORK

Hierarchical Control with LLMs The proliferation of LLM technology, coupled with their capability to interpret user prompts and perform reasoning, has led to growing interest in utilizing LLMs for robotics [14], [15]. Of particular notice and relevance are the use of LLMs for high-level reasoning in hierarchical control architectures. Prior work has demonstrated this by leveraging the few-shot prompt capabilities of LLMs [9], [8], their ability to code and compose functions [10], [16], or their ability to interact with human users through language [17]. In contrast to these works that attempt to use LLMs "as-is" and compose low-level skills, our work performs end-to-end fine-tuning through learnable latent codes. This includes finetuning some layers of the LLM through LoRA[18]. Empirically we show that such finetuning can outperform methods that use LLMs out-of-the-box.

Language Conditioned Imitation Learning To leverage LLMs for task planning and reasoning, such models need to be able to call preexisting lower-level skills to affect change in the environment. This can be achieve in two ways: (a) by leveraging *semantics* of the skills through language descriptions (e.g. go-to, reach etc.) as described above; or alternatively (b) through language conditioned policies which accept a text description as input to directly produce an action [12], [19], [1], [20], [21]. Such policies can typically perform only short horizon tasks and lack the reasoning and planning capabilities often found in LLMs. Our goal in this work is to leverage such "simple" or "primitive" language-conditioned policies along with LLMs to enable a hierarchical system to perform complex tasks that require multi-step planning and reasoning.

Large Pre-Trained Models for Embodied Agents Recent years have witnessed growing interest in robotics to re-use large models originally trained for vision or language applications [22], [15] or their architectures [23], [24], [25], [26], [27]. We are also starting to see large models and representations custom trained for robotics [1], [28], [29], [30]. In our work, we leverage the recent class of Multimodal Large Language Models [31], [32], [33] that extend the capability of text only LLMs to interpret other modalities like vision through alignment layers. Specifically, our instantiation of LCB model builds on top of LLaVA [31] and finetunes the model on a simulated dataset of embodied reasoning and long-horizon tasks. As the availability of embodied datasets paired with language annotations grow, we hope that our method can be extended to release generalist models that can be deployed zero shot in new domains.

III. METHOD

We wish to develop a hierarchical policy architecture that can enable robots to perform a variety of manipulation tasks when provided with free-form language descriptions. Specifically, we seek an architecture that can handle low-level actions for fine-grained or contact-rich tasks (e.g. pushing, 6D object manipulation) while also having the capability to reason and plan without any external step-by-step instructions. Before we present our architecture for this purpose, we first survey two other families of approaches and their deficiencies, which provides the intuition and basis for our method. These approaches are shown in Figure 2.

LLMs Leveraging Predefined Skills First, we can consider a hierarchical approach where LLMs perform high-level task planning by calling a set of pre-defined skills or APIs [8], [10]. These lower level skills (e.g. go-to, push) are described and provided to the LLM as part of the main text prompt. This approach suffers from three primary drawbacks. First, for an LLM to plan with skills, they need to have *semantics* attached to them that make linguistic sense. Second, this constrains the set of skills to a closed vocabulary and prevents any form of generalization to new skills or capabilities. Last, tuning those predefined skill libraries can be challenging. If the provided skills are too primitive (e.g.,

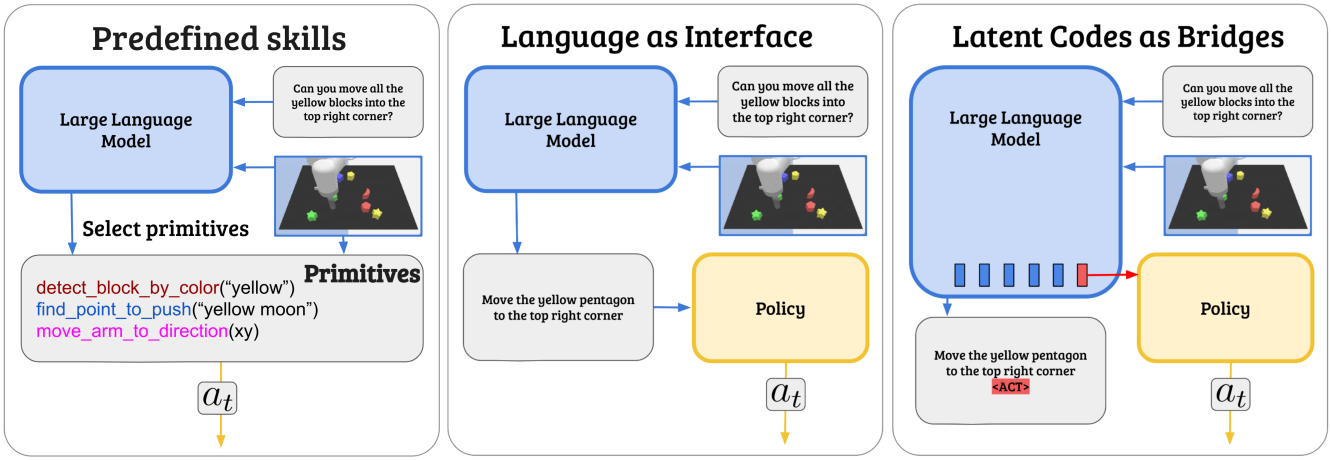


Fig. 2: A high level architectural comparison of LLM-based hierarchical policies. Predefined skills (left) uses a LLM to call predefined primitives. Language as an interface (middle) uses a LLM to output a simple language command, which is then passed into a language conditioned policy. **LCB** (right) utilizes a latent code as a **bridge** between the LLM and the low level policy, facilitating hierarchical control and end-to-end learning.

joint angle, end-effector rotation), the LLM may struggle to construct executable action sequences. Conversely, if the skills are too high-level, the range of tasks the robot can complete will be limited. Furthermore, code-writing proficiency demands a high-quality LLM, a criterion met chiefly by proprietary commercial models such as GPT-4 [2]. Additionally, end-to-end fine-tuning is challenging since the LLM cannot adapt or compensate for limited prowess of the low-level skills [8].

Language as Interface The second class of approaches can leverage *language-conditioned low-level policies* as opposed to a finite set of low-level skills. Such policies can take a simple language command as input (e.g. `pickup the red block`) and produce actions that can (hopefully) accomplish the task. Since these policies can accept free-form text as input, at least theoretically, they have the capability to generalize to new instructions. Furthermore, they are amenable to end-to-end fine-tuning from high-level instructions, through an LLM, to the language conditioned policy, and ultimately the action. Nevertheless, this class of approaches also suffer from key limitations. First, not all high level tasks can be decomposed into sub-tasks in simple language. For example, imagine trying to describe step-by-step instructions to make a robot dance to a song. Second, end-to-end fine-tuning with such an architecture can erase planning and reasoning capabilities that the LLM originally had [34].

Latent Codes as a Bridge (Ours) Finally, we describe our method which can overcome the key limitations outlined above. Our key insight is that we can introduce an additional latent code to act as a bridge between the high-level LLM and low-level language conditioned policy. We augment the LLM's tokenizer by adding a specialized `<ACT>` token, prompting the model to predict this token in response to actionable questions. The last layer embedding of the `<ACT>` token is then utilized as a latent goal for the downstream policy network. This learnable `<ACT>` token's embedding facilitates the transmission of abstract goals and nuances to the low-level policy – details that are not easily conveyed through language alone. Furthermore, by using this additional learnable token, we preserve the embedding space for language tokens, thus preventing any catastrophic forgetting during end-to-end fine-tuning. We describe more specific details of our architecture and implementation below.

A. *Architecture and Implementation Details of* `LCB`

`LCB` unifies the capabilities of a slow but powerful pretrained Multimodal Large Language Models (LLMs) with a fast and simpler decision-making policies to create a model that ingests vision and language inputs to output low-level actions. This integration involves a two-component system: a pretrained LLM, denoted as f_ϕ, and a pretrained policy, π_θ, parameterized by ϕ and θ respectively. The LLM consists of a text only large language model and a vision encoder, which projects images into the text only large language models embedding space, facilitating a multimodal understanding of textual and visual inputs. In this work, we leverage LLaVA[31] as our pretrained LLM. f_ϕ takes in text tokens x_{txt} and images x_{img} and outputs text tokens. The pretrained policy π_θ takes as input environment observations at the current time step o_t, with conditioning latent z, and outputs the action at the current time step a_t.

We introduce an additional `<ACT>` token into the vocabulary of the language model, which is a special token that enables the language model to generate an action embedding to control the lower level policy. The model is trained to output `<ACT>` tokens when executable requests are provided to the model. We extract out the last-layer embedding features from the model of at the `<ACT>` token, following the approach used in Language Instructed Segmentation Assistant (LISA)

[35]. This embedding is projected into the policy latent conditioning space by a linear layer to extract the latent feature $z_{\texttt{<ACT>}}$ which is then fed into the policy π_θ.

B. Data Processing

The LCB framework necessitates diverse and strategically curated datasets to make the policy effective for language-guided action execution in varied contexts. We cater the data collection and preprocessing steps towards this goal, creating a small instruction tuning dataset.

We convert in-domain text conditioned policy data into the chat format of LLM assistants. Typical language conditioned trajectory datasets contain one language instruction and a list of (observation, action) pairs $[x_{txt}, (o_0, a_0, ..., o_t, a_t, ...)]$ per trajectory. We programmatically generate text data in the format of chat interactions using templates. A simple example of this user-assistant interaction, is "User: can you help me x_{txt}? Assistant: yes, <ACT>." This trains the model to recognize and respond to direct action requests, fostering a conversational interface that seamlessly transitions from dialogue to action.

Moreover, we enrich our training material with additional datasets designed to prompt specific behaviors from the language model. One such data source is reasoning data, where the model is tasked with a more abstract goal and must reason about the scene to accomplish the goal. Such examples are framed within a chat-like interaction, encouraging the model to articulate its reasoning process before executing the <ACT> command. For example, "User: x_{img} Can you x_{txt}? Assistant: I will x_{goal} <ACT>". Where x_{txt} does not explicitly specify the target object and location. If x_{txt} is "move the block closest to the bottom right to the block of a similar color", the assistant's response, x_{goal}, provides an explanation of the task, such as "I will move the blue cube on the bottom right to the blue moon".

We also study long-horizon tasks and incorporate training sequences that require the model to plan and execute multiple steps to achieve a goal. This is achieved by defining task stages (start, regular, transition, stop) and incorporating the previous action as context in the language model's input. One example of such long-horizon tasks is: "User: x_{img} Can you sort the red blocks into the bottom left corner?" To solve this task, the model needs to understand which blocks are red and come up with a plan to break this long-horizon task down into two trajectories by first moving the red pentagon and next moving the red moon. Additionally, the model should understand if the subtask is finished and decide if it needs to switch to the next subtask to complete the long-horizon task or keep doing the previous task. We manually set the beginning of this long-horizon task as the "start" stage. We prompt the model with the high-level task description and let it initiate the first subtask. Timesteps within one subtask are labeled as the "regular" stage, where the model needs to know that it is still working on the current subtask and should do so until it is complete. We define the "transition" stage at the end of one subtask and the start of the next subtask. The model should be able to recognize that the previous task is finished and start working on the next subtask. At the "end" stage when the long-horizon task is done, we prompt the model to conclude the task and predict null actions to ensure the policy won't keep doing random movements. Comprehensive prompts for all for stages are auto-generated through a scripting process. This strategy trains the model to recognize task progression and adapt its actions accordingly, enabling it to manage tasks with evolving objectives. Through this dataset strategy, our model is finely tuned as a versatile tool capable of understanding and executing a wide range of language-guided actions.

We collected 400 trajectories for each reasoning task and 1200 trajectories for each long horizon task. We use an oracle scripted policy [21] to generate the data automatically. The oracle is provided the underlying ground truth target object and target location, which can be extracted from the simulation state.

C. Training

The training of LCB employs a combination of techniques to integrate the LLM and policy components. We leverage Low Rank Adaptation [18] (LoRA) for fine-tuning the LLM, allowing for more efficient training. We adopt a cold start approach to policy training, reminiscent of staged training strategies seen in prior works, by first freezing the action decoder and only fine-tuning the language model. This preliminary phase focuses on aligning the embeddings produced by the LLM with the feature space of the policy, to prevent the initial unstable gradient due to the mismatch between the pre-trained policy and the pre-trained LLM. We find that adding an additional CLIP loss to regularize the latent embedding $z_{\texttt{<ACT>}}$ is necessary, ensuring that the embeddings from the language model remain well aligned with the lower

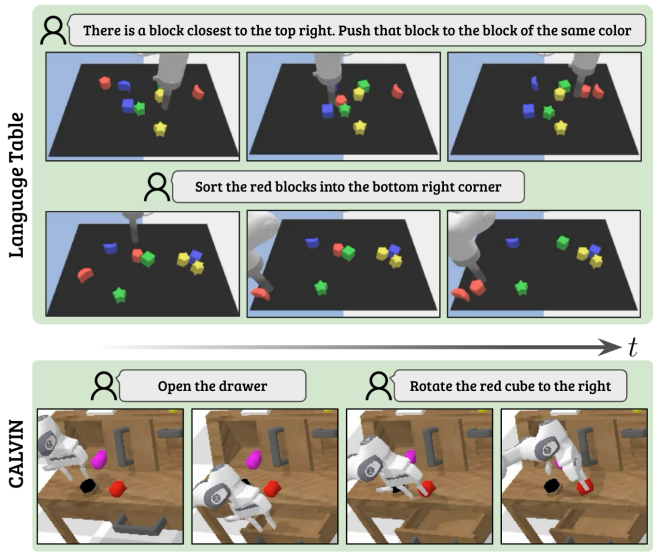
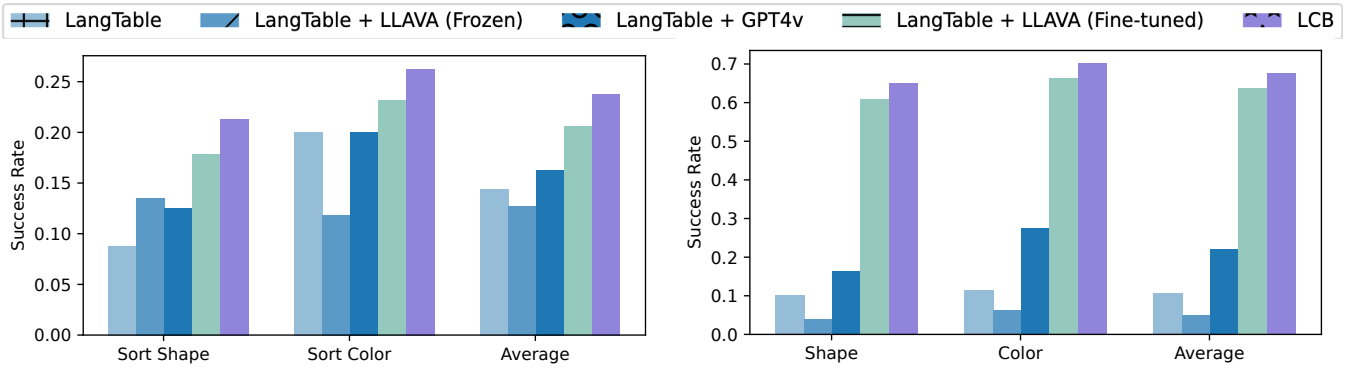


Fig. 3: A visualization of the two environments along with exemplar tasks that we train and evaluate on. The top depicts the Language Table environment [12]. We study reasoning tasks (first trajectory) and long horizon tasks (second trajectory). The bottom depicts the CALVIN long horizon benchmark [13], in which the agent must sequentially accomplish tasks.



(a) Long Horizon Success rate for the multi-step tasks on Language Table. The task requires shorting some blocks based on color or shape in a given direction. The environment only provides the high level objective to each method. This task requires the policy to have more long term planning capabilities, whether explicitly or implicitly.

(b) Reasoning: Success rate for the reasoning tasks on Language Table. The reasoning task is specified as a variant of "There is a block that is closest to i.e., top right corner. Push that block to the other block of the same shape/color." This task requires the agent to understand object semantics and spacial relationships.

Fig. 4: Task success rates on Language Table. The tasks are drawn from the higher level Language Table tasks from PALM-E [37]. LangTable refers to the original language table policy [12]. +LLaVA (frozen) refers to composing the original language table with a frozen LLaVA model and few shot prompting. +GPT-4V similarly refers to composing the original policy with GPT-4V. +LLaVA (finetuned) refers to finetuning the LLaVA policy on our mixture dataset on the language only, then composing it with the policy. Our results show that leveraging LCB is effective on tasks that require additional reasoning and planning. Note that the same model is evaluated between the long horizon and reasoning tasks.

level ground truth text description g_{txt} of the objective for the pre-trained policy. In total, our loss function is comprised of 3 terms, and can be expressed as follows:

$$\mathscr{L} = \lambda_1 \mathscr{L}_{\text{policy}}(\pi_\theta, o_t, a_t, z_{\texttt{<ACT>}}) \quad (1)$$
$$+\lambda_2 \mathscr{L}_{\text{LM}}(f_\phi, x_{\text{txt}}, x_{\text{img}}) \quad (2)$$
$$+\lambda_3 \mathscr{L}_{\text{CLIP}}(z_{\texttt{<ACT>}}, g_{\text{txt}}) \quad (3)$$

$\mathscr{L}_{\text{policy}}(\pi_\theta, o_t, a_t, z_{\texttt{<ACT>}})$ depends on the design of the pre-trained policy network and can be anything in principle. For the LCB LanguageTable setup, since we inherit the pre-trained policy network used in [21], the policy loss here is the L2 loss. For the LCB CALVIN setup, we use the 3D diffuser Actor [36] policy, where the policy loss is the diffusion denoise loss. In fact, LCB is compatible with any policy loss, as long as the policy takes the <ACT> embedding as the input. $\mathscr{L}_{\text{LM}}(f_\phi, x_{\text{txt}}, x_{\text{img}})$ is the same auto-regressive training objective following [35], with a special token <ACT> instead. When the answer is actionable, the model must predict the <ACT> token as the answer, as specified by our dataset. We use a frozen pre-trained CLIP text model to make sure the LCB output action embedding won't be too off from the CLIP embedding. Specifically we employ, $\mathscr{L}_{\text{CLIP}}(z_{\text{act}}, g_{\text{txt}}) = \cos(\text{stop\_gradient}(\text{clip}(g_{\text{txt}})), z_{\text{act}})$. This auxiliary loss helps regularize the predicted embedding.

During training, we use LoRA with a rank of 16. All the reported results use LLaVA with Llama2 7B [7] as the underlying text LLM. Training takes about 8 hours to finish on an 8 80GB A100 GPU DGX machine.

IV. Results

We systematically evaluated LCB across a diverse set of environments and tasks to demonstrate the efficacy of integrating a pretrained Large Language Model (LLM) with a domain-specific, pretrained low-level policy. Our primary objective is to study the capabilities of the policy, specifically its high-level language understanding and low-level control. Through our experiments, we aim to answer the following questions:

- Does LCB enable learning a bridge between the LLM and the policy more effectively than pure language?
- Can LCB leverage the pretrained capabilities of LLMs to solve long horizon tasks by decomposing the high level goals into the step by step latent commands?
- Can LCB outperform other baseline methods that leverage close-sourced state of the art LLMs such as GPT-4V?

To answer these questions, we study how LCB performs under various reasoning and long horizon settings in both the Language Table and CALVIN benchmarks. See Figure 3 for a visualization of the environments and example tasks.

A. Evaluation on Language Table

Language Table offers a simulated tabletop environment for executing language-conditioned manipulation tasks [12]. The environment features a flat surface populated with blocks of various colors and shapes, alongside a robot with a 2D action space. Language Table provides observations in the form of the robot end-effector position and third-person camera images. Despite its simplicity, it provides a reproducible and comprehensive environment to study challenges at the interface of high level language and low level contact-rich dynamics and feedback control.

We investigate the benefit of using LCB on the original Language Table benchmark. Here we apply our method using the same dataset that the original Language Table model was trained on, translating the original language instructions into

chat interactions with action tokens as specified in Section III. As shown in Table I, with the end to end optimization with the pretrained LLM, the success rate across the benchmark matches or exceeds the baseline Language Table approach. This signifies that LCB is able to seamlessly adapt a pretrained LLM and policy together. We suspect that this is due to the flexibility in the latent representation $z_{\texttt{<ACT>}}$, allowed for by our approach as well as additional capacity afforded my the language model.

We next investigate more complex language tasks that require reasoning and planning capabilities. We collect a small dataset for each capability, training models to compare the following approaches:

- **LangTable:** The original Language Table Policy, as provided by [12].
- **LangTable + LLaVA (Frozen):** The combination of the original policy and a non-fine-tuned LLaVA model interfacing through language. We prompt LLaVA to output language commands in the format and style as expected by LangTable.
- **LangTable + GPT-4V:** The integration of LangTable with the state-of-the-art proprietary Vision Language Model (GPT-4V). In order to bootstrap the spatial understanding of GPT-4V, we also incorporate the Set of Marker (SOM) [38] technique to enhance the GPT-4V's capability. We further include multi-modal few show contexts including language explanation of the tasks and image examples. We tune and experiment with the prompting strategy to maximize performance.
- **LangTable + LLaVA (Fine-tuned):** The original policy augmented by a LLaVA model that has been fine-tuned on the exact language needed for the action policy for the given task.
- **LCB:** We take a pretrained LLaVA model and the pre-trained LangTable policy and apply LCB, learning a latent interface between the two on the respective instruction dataset.

Results for long horizon performance are provided in Figure 4a. In this task, the agent must sort blocks based on shape or color into a specified corner of the board, requiring a long sequence of actions from which the agent could greatly benefit through high-level planning. We see that LCB exhibits a competency for handling such tasks, as indicated by the heightened success rates, improving on pure language interface baselines. This is attributable to the method's ability to generate a coherent sequence of latent action embeddings that guide the policy through the task's duration, facilitating a more consistent and accurate alignment with the sequential nature of the task. During evaluation, we run the higher level language model at a slower rate than the lower level policy, only updating the language models output every 40 environment steps. While running the higher level language model more frequently may possibly yield better results, we found that for this task, there is no significant benefit to running the high level model at a faster frequency. In between the update cycles of the language model, the low-level policy continually runs based on the latest latent action embedding. The environment will give an episode end signal when the goal is reached, following the setup in Language Table [21]. We find that this approach increases computational efficiency without compromising task performance suggesting the effectiveness of the model hierarchy.

Results for reasoning performance are provided in Figure 4b. Tasks here are of the form "There is a block that is closest to {corner}. Push that block to the other block of the same {shape/color}". In order to successfully accomplish this task, the agent must identify which block is located closest to a given corner, identify the relevant property (i.e. shape or color) and consolidate that understanding into an executable instruction. We see that our approach is able to outperform baselines that involve zero-shot prompting as well as naively fine-tuning the language model to output the translated robot task. We see that fine-tuning the language model to output the ground truth language primitive is effective in reaching parity with the oracle language baseline, but that LCB is able to match and even exceed that.

We provide a qualitative assessment of the language output from the various top performing approaches in Figure 5. LangTable + GPT-4V requires heavy prompt engineering and additional string parsing to extract out the final policy. LangTable + LLaVA is effectively fine-tuned by outputting the direct low level text command to the policy, but no longer is able to maintain a chat like interface to the user. In contrast, LCB is able to output an effective embedding for the low level policy while also verbalizing its reasoning. This decouples the low level policy conditioning from the language models text outputs, offering increased flexibility during instruction fine-tuning.

TABLE I: Comparison on the original Language Table benchmark tasks. LangTable is the original language table policy [12]. LCB is our method applied only to the original Language Table dataset. We see that LCB can help improve task performance by leveraging the vision language model for feature extraction. The tasks are: Block to Block (B2B), Block to Block Relative Location (B2RL), Seprate (S), Block to Relative Location (B2RL), and Block to Absolute Location (B2AL).

Model	B2B	B2BRL	S	B2RL	B2AL	*Avg*
LangTable	0.88	0.70	0.94	0.68	0.65	0.77
LCB	0.90	0.66	0.99	0.73	0.71	**0.80**

B. Evaluation on CALVIN

CALVIN[13] is an open-source simulated benchmark designed for learning long-horizon tasks conditioned by language. The environment features a 7-DOF Franka Emika Panda robotic arm equipped with a parallel gripper, situated at a desk with a variety of articulated furniture and objects for interaction. In each experiment, the robot needs to solve a sequence of complex full 6D manipulation tasks governed by real-world physics and guided by a series of language instructions. Each subtask is paired by a specific language instruction; upon successful completion, the robot proceeds to the next subtask accompanied by a new instruction. CALVIN

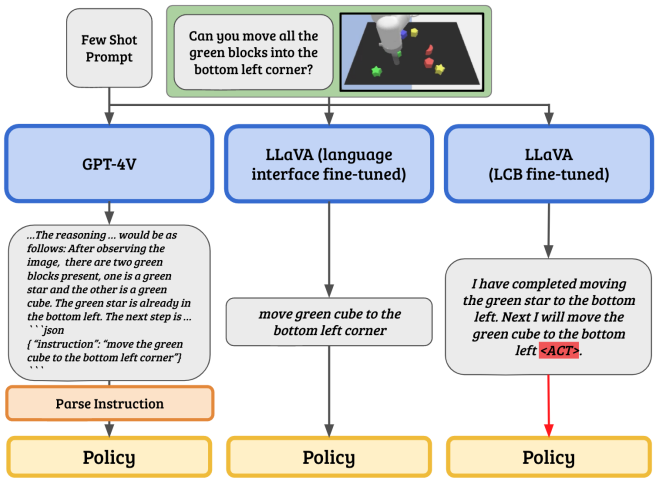


Fig. 5: A comparison of the flow from a high level language task to the policy for different approaches. **(Left) LangTable + GPT-4V** requires a prompt to understand the task and desired output format. GPT-4V can provide language reasoning to allow the user to introspect the decision process of the language model, but requires additional parsing to extract the relevant language instruction to provide to the model. **(Middle) LangTable + LLaVA (Fine-tuned)** fine-tunes the language model to output the exact language instruction as in the training data, effectively acting as a language interface converter. This approach, while effective, removes the chat like capability from the language model. **(Right) LCB** fine-tunes the language model with a chat like interface and action token. The policy is directly conditioned on the latent feature from the action token provided by the model, enabling effective policy conditioning without losing the chat like language model interface.

encompasses four distinct environments A, B, C and D, with a shared set of language instructions and subtasks.

In order to demonstrate the generalization capabilities of LCB across various environments as well as its ability to comprehend and act upon the same instructions phrased differently in the CALVIN long horizon full 6D manipulation setting, we compare the following approaches:

- **RoboFlamingo (RF):** RoboFlamingo[39] adapts OpenFlamingo[40] by fine-tuning solely the cross-attention layer to directly output actions, thus maintaining its language comprehension. However, this approach requires executing the entire LLM anew with each progression to a subsequent state, leading to inefficiencies.
- **3D Diffusion Actor (3DDA):** Incorporating a diffusion policy with 3D scene representation and CLIP[41] language embedding, the 3D Diffusion Actor [36] sets the current SOTA on the Calvin benchmark when provided with standard language instruction inputs. However, a notable limitation stems from the constraints of the CLIP text model it employs. 3DDA can not generalize well on language instruction outside of its training distribution.
- **LCB:** LCB for Calvin integrates a pre-trained LLaVA[31] as the Multimodal Large Language Model backbone with a pre-trained 3D Diffusion Actor serving as the action policy. This combination leverages the SOTA capabilities of the 3D Diffusion Actor to achieve a synergistic effect: LCB for Calvin excels in both language comprehension and low-level manipulation. Since RoboFlamingo runs the entire LLM on every environment step, in order to make a fair comparison, we also run the LLM part of LCB synchronously with the downstream policy, although we notice no significant performance difference for Calvin.

TABLE II: Task completion rates for various methods on CALVIN[13] long-horizon tasks. All methods were trained exclusively on the ABC split of Calvin with the original language annotations and tested on split D with GPT-4 enriched language annotations, following the RoboFlamingo enriched instruction evaluation setting[11]. *RF denotes our own training of the RoboFlamingo model on the ABC Calvin split. 3DDA denotes the policy from 3D Diffuser Actor [36].

	Model	RF[39]	3DDA[36]	**LCB**
Task Completed in a Sequence (Success Rate)	1/5	0.620	0.652	**0.736**
	2/5	0.330	0.391	**0.502**
	3/5	0.164	0.203	**0.285**
	4/5	0.086	0.117	**0.160**
	5/5	0.046	0.061	**0.099**
	Avg Len	0.40	1.42	**1.78**

Table II presents results for the CALVIN long-horizon, language-conditioned benchmark. In this setting, the robot executes a series of tasks in unfamiliar environments based on novel GPT-4 enriched [40] instructions not encountered during training. The experimental outcomes demonstrate our approach's distinct advantage over baseline methods. LCB significantly surpasses all baselines in terms of task success rate at every stage and in average completed trajectory length.

V. Conclusion

In this work, we introduce a novel approach, Latent Codes as Bridges, or LCB, that combines the abstract reasoning capabilities of large language models with low-level action policies. Our methodology does not merely stack these capabilities as in prior works but integrates them in an end-to-end fashion through a learned latent interface. The empirical evidence from our evaluations on the Language Table and CALVIN benchmarks shows the model's adeptness in interpreting and executing various reasoning and long horizon objectives. The flexibility and effectiveness of the hierarchy enabled by LCB shows promise for real world robotic applications.

Acknowledgments

Yide Shentu is supported in part by InnoHK Centre for Logistics Robotics and ONR MURI N00014-22-1-2773. Philipp Wu is supported in part by the NSF Graduate Research Fellowship Program. We thank Xinyang Geng and Fangchen Liu for valuable discussions regarding LCB.

REFERENCES

[1] A. B. et al., "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023.

[2] F. Liu, K. Fang, P. Abbeel, and S. Levine, "Moka: Open-vocabulary robotic manipulation through mark-based visual prompting," 2024.

[3] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor, "Language-conditioned imitation learning for robot manipulation tasks," 2020. [Online]. Available: https://arxiv.org/abs/2010.12083

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, pp. 39:1–39:40, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:7242892

[5] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," 2022. [Online]. Available: https://arxiv.org/abs/2204.06252

[6] OpenAI, "Gpt-4 technical report," 2023.

[7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[8] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as I can, not as I say: Grounding language in robotic affordances," Apr. 2022.

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

[10] 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," Sept. 2022.

[11] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Ok-robot: What really matters in integrating open-knowledge models for robotics," 2024.

[12] 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," Oct. 2022.

[13] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," 2022.

[14] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," *ArXiv*, vol. abs/2311.07226, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265149884

[15] S. Vemprala, R. Bonatti, A. F. C. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," *ArXiv*, vol. abs/2306.17582, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259141622

[16] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

[17] B. Li, P. Wu, P. Abbeel, and J. Malik, "Interactive task planning with language models," 2023.

[18] 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," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[19] 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," *ArXiv*, vol. abs/2202.02005, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:237257594

[20] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong, "Vision-language foundation models as effective robot imitators," *arXiv preprint arXiv:2311.01378*, 2023.

[21] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," *Robotics: Science and Systems*, 2021.

[22] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. B. Girshick, "Segment anything," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257952310

[23] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Neural Information Processing Systems*, 2021.

[24] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, 2021.

[25] P. Wu, A. Majumdar, K. Stone, Y. Lin, I. Mordatch, P. Abbeel, and A. Rajeswaran, "Masked trajectory models for prediction, representation, and control," in *International Conference on Machine Learning*, 2023.

[26] F. Liu, H. Liu, A. Grover, and P. Abbeel, "Masked autoencoding for scalable and generalizable decision making," 2023.

[27] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel, "Any-point trajectory modeling for policy learning," 2024. [Online]. Available: https://arxiv.org/abs/2401.00025

[28] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *Conference on Robot Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247618840

[29] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, P. Abbeel, J. Malik, D. Batra, Y. Lin, O. Maksymets, A. Rajeswaran, and F. Meier, "Where are we in the search for an artificial visual cortex for embodied intelligence?" 2023.

[30] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. Kaelbling, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," 2024.

[31] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[32] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv preprint arXiv:2304.10592*, 2023.

[33] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International Conference on Machine Learning*, 2023.

[34] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, "An empirical study of catastrophic forgetting in large language models during continual fine-tuning," 2023.

[35] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia, "Lisa: Reasoning segmentation via large language model," *arXiv preprint arXiv:2308.00692*, 2023.

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

[37] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "PaLM-E: An embodied multimodal language model," Mar. 2023.

[38] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," 2023.

[39] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong, "Vision-language foundation models as effective robot imitators," 2024.

[40] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, S. Sagawa, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt, "Openflamingo: An open-source framework for training large autoregressive vision-language models," 2023.

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.