

Language Movement Primitives: Grounding Language Models in Robot Motion

Yinlong Dai¹, Benjamin A. Christie¹, Daniel J. Evans¹, Dylan P. Losey¹, and Simon Stepputtis²

Abstract—Enabling robots to perform novel manipulation tasks from natural language instructions remains a fundamental challenge in robotics, despite significant progress in generalized problem solving with foundational models. Large vision and language models (VLMs) are capable of processing high-dimensional input data for visual scene and language understanding, as well as decomposing tasks into a sequence of logical steps; however, they struggle to ground those steps in embodied robot motion. On the other hand, robotics foundation models output action commands, but require in-domain fine-tuning or experience before they are able to perform novel tasks successfully. At its core, there still remains the fundamental challenge of connecting abstract task reasoning with low-level motion control. To address this disconnect, we propose Language Movement Primitives (LMPs), a framework that grounds VLM reasoning in Dynamic Movement Primitive (DMP) parameterization. Our key insight is that DMPs provide a small number of interpretable parameters, and VLMs can set these parameters to specify diverse, continuous, and stable trajectories. Put another way: VLMs can reason over free-form natural language task descriptions, and semantically ground their desired motions into DMPs — bridging the gap between high-level task reasoning and low-level position and velocity control. Building on this combination of VLMs and DMPs, we formulate our LMP pipeline for zero-shot robot manipulation that effectively completes tabletop manipulation problems by generating a sequence of DMP motions. Across 31 real-world manipulation tasks, we show that LMP achieves 65% task success as compared to 35% for the best performing baseline. See videos at our website: <https://collab.me.vt.edu/lmp>

Index Terms—Task and Motion Planning, Manipulation, Hybrid Learning, Neuro-Symbolic Method

I. INTRODUCTION

Robots should be able to perform new tasks given simple user instructions. For example, imagine that a human tells their robot to “clean the plate.” Ideally, the robot reasons over this specification and the observed environment, breaks the task down into steps, and then cleans the plate. What makes this particularly challenging is that the human’s command implies motions that are never explained: e.g., picking up a cloth and moving it in a circular pattern. Consider Figure 1 where the robot realizes that it needs a cleaning implement, grasps a sponge, and then starts wiping. Unlike prior works in which the robot relies on human demonstrations [1]–[3] or multiple rounds of real-world experience [4], [5], we achieve *zero-shot* generalized intelligence by connecting the human’s language specification to the robot’s controlled motions.

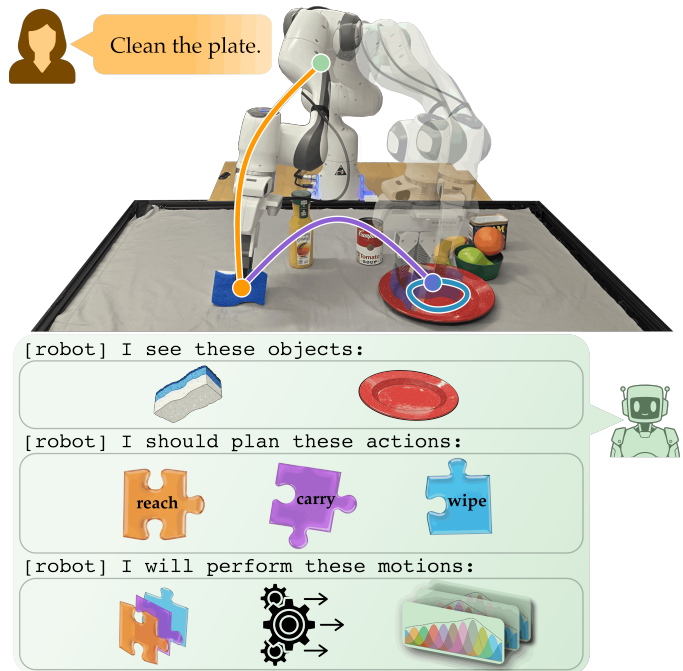


Fig. 1: Overview of Language Movement Primitives (LMP). Given a task description from the user, LMP first detects objects in the environment and composes a suitable sequence of high-level subtasks to achieve the overall goal. For each subtask, LMP then generates low-level parameters to define a Dynamic Movement Primitive (DMP). The robot tracks the continuous DMP trajectory, grounding its semantic reasoning for zero-shot robot manipulation.

Our work is related to recent efforts at the intersection of robotics and foundation models and view these efforts along a spectrum. At one extreme are methods that train models specifically on large-scale *robotic datasets*: these approaches obtain a policy that can directly convert language and vision into actions for the robot to execute [3], [6]–[8]. At the other extreme are methods that train on larger and more diverse *non-robotic datasets* [9], [10] with the goal of creating general-purpose foundation models, allowing these generalized vision and language models to decompose tasks and output actions at a symbolic level (e.g., “pick up the sponge”). However, both approaches have critical shortcomings. Robotics foundation models, while trained on extensive robot data, have limited common-sense reasoning capabilities and still require in-domain fine-tuning [8] or experience collection [11]. General purpose foundation models are not grounded in robotics, and thus their outputs are not immediately actionable — while the

This work is supported in part by NSF Grant #2337884.

¹Collab, Dept. of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061. {daiyinlong, benc00, daniellevans, losey}@vt.edu

²TEA Lab, Dept. of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061. stepputtis@vt.edu

robot knows *what* to do, it does not know *how* to “pick up the sponge” with respect to the robot’s joint’s actuation.

Recent neuro-symbolic approaches try to sidestep this gap by separating high-level task reasoning from low-level robot control. In recent works like [12]–[15] the robot combines a high-level planner — which provides generalized reasoning — with a low-level controller — which converts symbolic plans into robotic motions. However, these approaches perform symbolic reasoning over *discrete* action primitives rather than the *continuous* motion parameters, constraining the system’s ability to perform complex movements. This means our fundamental bottleneck persists: to “clean the plate,” we must establish a robust connection between high-level task reasoning and low-level robotic control for precise position, velocity, and acceleration profiles [16], [17].

In this work we propose to bridge the gap and provide generalized robot policies by connecting vision and language models (VLMs) to Dynamic Movement Primitives (DMPs). We consider DMPs to be a particularly suitable controller formulation for grounding language into robot motion, as they combine semantically meaningful parameters that define the motion, i.e., their forcing function, while providing stable goal-attractor dynamics. We specifically focus on table-top manipulation tasks. Given a user command (e.g., “clean the plate”, see Fig. 1), the VLM outputs task-based reasoning, breaking down the steps and even realizing what is missing from the human’s specification. DMPs offer a control-theoretic way to convert each of these steps into motions, parameterizing trajectories while guaranteeing the robot will reach its goal. Our insight is that:

DMPs physically ground VLMs because they embed diverse and expressive trajectories into a small number of parameters which the VLM can intuitively tune.

Our resulting system — which we refer to as **Language Movement Primitives (LMPs)** — integrates off-the-shelf VLMs into a full-stack pipeline for robotic task execution, combining the best of both worlds by leveraging the high-level reasoning capabilities of VLMs with the precise control of DMPs. During inference, LMP uses the human’s task description and an RGB-D image of the robot’s environment. By segmenting the input image, we generate a language description of the environment, including object types and locations. The VLM then reasons over this verbal state to create a high-level plan, and decides on the next subtask towards completing that plan. More specifically, the VLM chooses the parameters for the next DMP, and the robot executes this DMP in the environment. If a failure occurs, we incorporate a refinement loop where the human can provide corrective feedback (e.g., “use a towel instead”), and the robot iteratively refines its motion controller for subsequent attempts.

Overall, we see this paradigm as a step towards general-purpose robot arms that can be controlled by everyday humans without requiring additional demonstrations. We make the following contributions:

Formalizing LMPs. We frame Language Movement Primitives as an abstracted policy. The *state* is the image and generated text description of the environment, and the *action* is the

weights of the next DMP. The human’s task specification is an implicit reward that the policy can leverage to choose actions, providing a semantically meaningful connection between high-level VLM planning and low-level DMP control.

Grounding Language to Control. We present a complete system that translates open-form user instructions and corrective feedback into fine-grained and stable low-level motion controllers. Our user interface accepts commands specified in natural language, and also supports effortless and intuitive user feedback without requiring detailed knowledge of robot kinematics or low-level control specifications.

Comparing to Foundation Models. We compare LMPs to foundation models, such as $\pi_{0.5}$ [18], in robotics and other neuro-symbolic approaches across 31 real-world manipulation tasks. Without any fine-tuning or feedback, LMP outperforms the best baseline by 19% (54% vs. 35%); with up to three rounds of natural-language feedback, the gap grows to 30%. As part of our tests, we evaluate on parts of the CALVIN benchmark in simulation, as well as 31 real-world tabletop manipulation tasks across static and dynamically changing environments. Performance particularly improves in scenarios that require trajectory shaping for obstacle avoidance or multi-stage tasks.

II. RELATED WORKS

Learning general-purpose robot policies remains an open challenge. Recent approaches range from small expert models tailored on specific robots and tasks [19]–[21] to large foundation models for cross-embodiment [8], [22]. Complementing these data-driven methods, neuro-symbolic approaches combine LLM reasoning with planning and motion control through symbolic representations [23], [24]. In the following subsections, we discuss these three approaches in more detail while positioning our proposed method alongside these approaches.

A. Small Expert Models for Robot Learning

Control-theoretic approaches such as [25] and [26] learn behaviors for particular tasks and robots with only a few examples, but struggle beyond their training distribution. Deep learning has demonstrated strong results for robot policy and multi-task learning by modeling complex action distributions [1], [27], whether through imitation [28], [29] or reinforcement learning [30], [31]. While effective, demonstration-centric paradigms impose a substantial burden on the user to provide suitable examples [32], while reinforcement learning methods require large-scale training due to the complexity of real-world environments [33]. Hybrid approaches that initialize policies with imitation or reinforcement learning and subsequently refine them through experience [34]–[36] partially address this issue; however, generalization to novel scenario remains limited. World models offer an alternative by enabling robots to plan with scenario roll-outs [37], [38], but remain computationally expensive and scale poorly to high-dimensional real-world environments.

A core challenge across these approaches — particularly in household and tabletop manipulation — is the high variety

of tasks that need to be learned and the associated difficulty of obtaining sufficient training data [39], [40]. Open X-Embodiment [41] addresses this data scarcity challenge by aggregating datasets from different real-world robot tasks, allowing for positive transfer across embodiments within similar task domains. In contrast, LMP leverages the *reasoning capabilities of large foundation models* to generate task-specific robot controllers through a semantically interpretable parameter space, *removing the need for demonstrations*.

B. Robotics Foundation Models

Internet-scale foundation models have enabled progress in robot task planning [42], multimodal perception [7], and zero-shot adaptation to novel tasks and environments [14], [43]. Specifically, Vision-Language-Action (VLA) models [7], [8] leverage vision and language pipelines to ground high-dimensional data in a shared embedding space, conditioning downstream action generation [21]. RT-2 [6] popularized this VLA paradigm by co-fine-tuning vision-language models on robotic trajectory data, enabling emergent capabilities such as chain-of-thought reasoning and semantic understanding that transfers from internet-scale pre-training. Other methods prompt language models to output code that generates trajectories [14], while related works leverage LLMs as evaluators or reward models to provide feedback for policy learning [13], [44]. Despite these advances, a fundamental gap remains in *grounding high-level semantic reasoning into low-level continuous control while generalizing across diverse tasks*. LMP addresses this gap by providing a semantically meaningful parameter space such that large-scale foundation models can effectively generate low-level controllers while performing common-sense task reasoning across the control parameters.

C. Hybrid Models

Neuro-symbolic and hybrid systems have shown success by leveraging symbolic reasoning and planning as an integral part of the robot’s decision making and motion generation process [12], [21], [42]. Recent approaches incorporate neural vision and language-based components into symbolic pipelines, enabling robots to reason over unstructured sensory inputs while maintaining structured task representations [23], [45]. For example, [46] employ LLMs as neuro-symbolic task planners compatible with standard planning approaches, applying their generative capabilities to overcome limitations of traditional symbolic planners in dynamic human-robot collaboration scenarios. Leveraging LLMs enables robots to perform complex task decompositions, semantic grounding, and symbolic planning, while relying on learned controllers for execution of atomic task primitives [13], [47]. However, *existing hybrid approaches typically rely on hand-designed symbolic abstractions or a discrete library of learned controllers that are tightly coupled to specific robots and tasks*.

Beyond task planning, structured motion representation through control-theoretic approaches, such as DMP, has been used to model robot behavior [26], [48], offering a foundation with convergence guarantees while modeling motion through learned forcing functions. Recent extensions have combined

DMPs with reinforcement learning [49], demonstrating their versatility for trajectory representation. Our LMP framework is inspired by these directions, leveraging a large language model for high-level reasoning, generating a set of parameters for a low-level DMP motion controller, bridging the gap between language understanding, common-sense reasoning, and fine-grained motion control.

Prior work, such as [50], uses LLMs to sequence predefined DMPs through symbolic pre/post-conditions, requiring the fitting DMPs to demonstrations of potential motions and object interactions. However, instead of sequencing existing policies, we propose to leverage the spatial reasoning capabilities of LLM to directly generate DMP weights and parameters, which is a largely unexplored problem. Our proposed method, LMP, is a demonstration-free zero-shot framework that directly generates DMP parameters from language and visual observations. Through our results, we show that LLMs find DMP weights semantically interpretable, while the expressive structure of DMPs enables these generated parameters to be physically grounded into executable robot motions. Furthermore, LMP introduces a unified pipeline of task decomposition, controller generation, and iterative refinement for robot motion generation.

III. PROBLEM STATEMENT

We consider settings where a robot arm is performing tabletop manipulation tasks. The human operator provides a natural language description of the task τ they want the robot to perform, and the robot converts this description into task execution (e.g., embodied motion). The robot does not have access to any in-domain demonstrations.

Observation. Let $o \in \mathcal{O}$ be the robot’s observation of its environment. In our experiments o contains the robot’s joint position (measured by onboard encoders) and an RGB-D image of the environment (taken using an external depth camera). This observation captures where the robot is, and also perceives the different objects on the table.

Policy. The robot converts the user’s task description $\tau \in \mathcal{T}$ and the environment observation o into actions:

$$a \sim \pi(o \mid \tau, o) \quad (1)$$

Here π is a *high-level* policy: the actions produced by this policy are not low-level position or joint torque commands. Instead, we treat the actions as motion primitives (with details below), and rely on the high-level policy π to select the next motion primitive that will help complete the given task τ .

Action. We define the robot’s actions $a \in \mathcal{A}$ as parameterized motions. Related neuro-symbolic works have instantiated actions as absolute positions [15], learned skills [42], or a library of general-purpose motions [51]. We instead use Dynamic Movement Primitives (DMPs) as the *actions* that connect vision and language models to embodied behaviors.

Dynamic Movement Primitives (DMP). DMPs parameterize a smooth motion between two points. Let ξ be a single degree of freedom in the robot’s end-effector space; e.g., ξ could be the robot’s position along the x axis. DMPs parameterize the

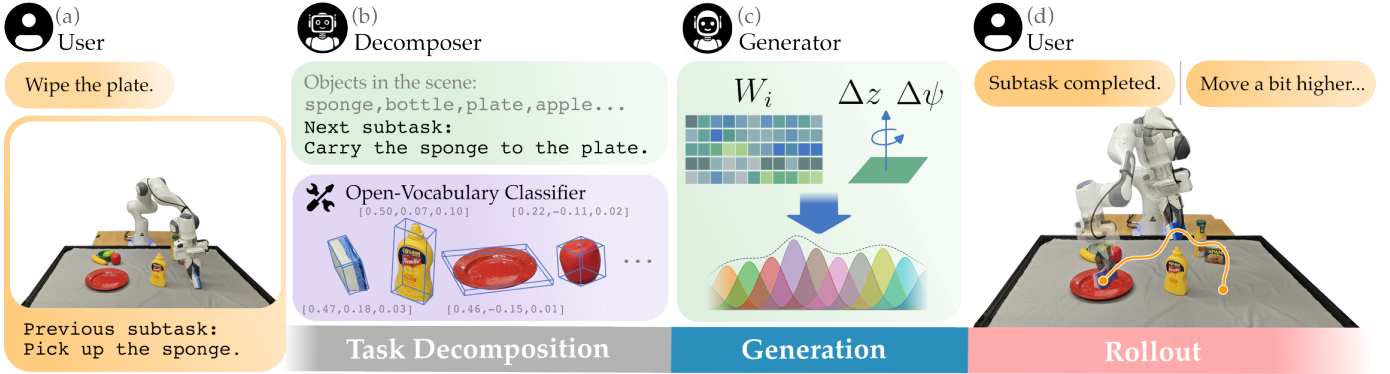


Fig. 2: LMP pipeline for a single subtask rollout. **(a)** The robot begins with a user-provided task description. The robot then collects an image capturing the current environment state, and remembers any previously performed subtask(s). **(b)** The decomposer $\pi_{\mathcal{D}}$ identifies scene objects and outputs a subtask for the next DMP to complete. An open-vocabulary classifier and depth sensing are used to estimate 3D object locations. The scene description and proposed subtask are then forwarded to the DMP weight generator $\pi_{\mathcal{G}}$. **(c)** The generator predicts DMP weights and auxiliary parameters to define the low-level reference trajectory. **(d)** The robot tracks the continuous trajectory generated from the predicted DMP parameters. Optionally, the user may observe the robot and provide natural-language feedback about any mistakes. If the user gives this refinement r , then the robot resets the rollout and the process repeats from (b).

trajectory $\xi(t)$ between the start state $\xi(0)$ and the goal state g as a second-order linear dynamical system [26], [52]:

$$T^2\ddot{\xi} = \alpha(\beta(g - \xi) - T\dot{\xi}) + f(z) \quad (2)$$

Here ξ is position, $\dot{\xi}$ is velocity, and $\ddot{\xi}$ is acceleration. The temporal scaling parameter T modulates the execution speed of the trajectory, while α and β are positive constants that define a spring-damper system (regulating how quickly the trajectory is attracted to the goal position g).

We can shape the trajectory of the DMP using the nonlinear forcing function $f(z)$. In our experiments we instantiate this forcing function as a normalized weighted sum of Gaussian basis functions $\{\psi_i\}_{i=1}^N$ parameterized by the phase variable \tilde{t} and modulated by the canonical decay variable $z \in [0, 1]$:

$$f(z) = \frac{z(t) \sum_{i=1}^N w_i \psi_i(\tilde{t})}{\sum_{i=1}^N \psi_i(\tilde{t})}, \quad \tilde{t} = \min\left(\frac{t}{T}, 1\right) \quad (3)$$

Each Gaussian basis function is given by:

$$\psi_i(\tilde{t}) = \exp(-h(\tilde{t} - c_i)^2) \quad (4)$$

where the basis centers c_i are uniformly distributed over $[0, 1]$ and $h = 1/\sigma^2$ controls the basis width.

Within our work the weights w are particularly important. Element w_i determines the relative impact of the i -th basis function; by increasing or decreasing these weights, we modulate the shape of trajectory $\xi(t)$ along a continuous spectrum. Indeed, we can specify the shape of the entire motion using weights w and goal position g . When using DMPs, the robot is guaranteed to converge to the goal as the canonical variable $z \rightarrow 0$. We update z as a time-dependent decay function of the normalized time \tilde{t} :

$$z(t) = \exp(-\gamma\tilde{t}^3) \quad (5)$$

where $\gamma > 0$ controls z 's rate of decay. In practice, z remains close to one until the end of the trajectory: when $z \rightarrow 1$, the

forcing function can alter the motion profile. As $z \rightarrow 0$, the nonlinear terms are ignored and the robot converges to g .

Overall, DMPs serve as motion primitives that capture a wide range of trajectories through a small number of weights w and goals g . We know that trajectory $\xi(t)$ will converge to the goal g , and directly adjust the trajectory shape using weights w . DMPs enable rich task-specific motion shaping without sacrificing stability. Within the context of our high-level policy in Equation (1), the set of DMP weights and goals becomes the robot's action space \mathcal{A} .

Subtasks. At the start of the interaction the human provides a task description τ . The robot then completes this task in a sequence of steps. At each step the robot observes o , and queries the high-level policy π for the weights and goals of the next DMP. The robot arm rolls out the DMP and interacts with the physical environment. This process then repeats until the task is complete (outputting a null command). We refer to the steps as *subtasks* τ_i , and denote the robot's observation and action at each subtask as o_i and a_i , respectively.

Feedback. Ideally, the sequence of actions $[a_i]_0^K$ should solve the original task τ . But we recognize that no policy is perfect and that the generated weights w may not complete the task. To this end, we incorporate additional natural language feedback at the end of each subtask, allowing the robot to re-attempt the task by refining the previously generated weights given verbal feedback. Alternatively, the human may have *new* instructions they want to provide. Our problem setting formulates this feedback as a natural language refinements r . We emphasize that this refinement is not necessary, but our framework incorporates it effectively when provided.

IV. LANGUAGE MOVEMENT PRIMITIVES (LMPs)

To convert high-level language prompts τ into low-level control trajectories we propose Language Movement Primitives (LMPs). The central idea of LMPs is a combination

Algorithm 1 Language Movement Primitives (LMPs)

Input: task prompt τ and DMP grounding prompt s_G

```

1:  $\Pi \leftarrow \emptyset$  ▷ Set of completed subtasks
2:  $c \leftarrow 0$  ▷ Retry counter for each subtask
3: repeat
4:    $o \leftarrow \text{environment\_state}$ 
5:    $\theta \leftarrow \pi_{\text{class}}(o \mid o)$  ▷ Extract object poses and labels
6:    $\varphi_i \leftarrow \pi_{\mathcal{D}}(o \mid \tau, o, \theta, \Pi)$  ▷ Decompose for next subtask
7:    $(W_i, \Delta z, \Delta \psi) \leftarrow \pi_{\mathcal{G}}(o \mid \varphi_i, s_G, o, \theta)$ 
▷ Generate DMP weights and goal offsets
8:    $\text{DMP}(W_i, \Delta z, \Delta \psi)$  ▷ Execute DMP in environment
9:    $r \leftarrow \text{judge}()$  ▷ Collect any feedback
10:  if  $r \neq \emptyset$  and  $c < 3$  then
11:     $\text{update\_prompts}(r)$  ▷ Add  $r$  to existing prompts
12:     $\text{reset}(\varphi_i)$  ▷ Reset to start of subtask  $i$ 
13:     $c \leftarrow c + 1$ 
14:  else
15:     $\Pi \leftarrow \Pi \cup \varphi_i$ 
16:     $c \leftarrow 0$  ▷ Reset retry counter
17: until  $\varphi_i = \text{done}$ 

```

of VLMs and DMPs. The VLM serves as the high-level policy from Equation (1): this model takes in segmented state observations, breaks the task down into steps, and decides on the next subtask. As an action, the VLM outputs the trajectory parameters a_i for the physical robot to execute. DMPs then map these parameters into a controlled robot motion: ensuring the robot will reach the given goal and providing a continuous reference trajectory. We hypothesize that this combination will be effective because DMPs capture a diverse set of complex motions through a small number of parameters, and that these parameters are semantically meaningful (e.g., a given weight might correspond to increased motion in the x axis).

In this section we present our LMP formalism. Our paradigm starts by generating a text summary of the state from image and depth observations (Section IV-A). Once the VLM is given this language information, it then *decomposes* the overall task into the next subtask the robot should perform (Section IV-B). Finally, the VLM *generates* the goals and weights for low-level DMPs that will achieve this subtask, and the robot executes the DMPs in its physical environment (Section IV-C). Across our experiments this pipeline was sufficient for completing most tasks on the first attempt. However, if the robot fails, we also incorporate a refinement step where the user can give natural language corrections, and the robot uses those suggestions to improve its next attempt (Section IV-D). Our overall method is summarized in Figure 1, Figure 2, and Algorithm 1.

A. From Observations to State Descriptions

The robot’s observations o include the robot’s joint position and RGB-D images of the environment. We directly provide the RGB environment image as an input to the VLM policy. However, we also translate our entire observation o into a templated natural language description of the state for the VLM to reason over. For example, “there is an [object label]

located at [this position] and oriented with [this orientation].” We achieve this automated segmentation by using an open-vocabulary classifier to identify objects in the environment:

$$\theta = [(l, p)_i]_0^N \sim \pi_{\text{class}}(o \mid o) \quad (6)$$

Here θ is the set of all segmented object information. This includes l , the textual object labels, and p , their 3D position and orientation in the robot’s coordinate frame. The policy π_{class} converts the observations into segmented data including the object’s label and global position in 3D space. In our experiments we use Gemini-Robotics ER [43] and LangSAM [53], [54], but other approaches for object detection and localization are also suitable [55]–[57]. Note that most off-the-shelf classifiers output pixel coordinates. Using the camera intrinsics, depth measurements, and the calibrated camera-to-robot extrinsics, we back-project pixels into 3D and transform them into the robot frame. Once θ is obtained, we can then use the information in θ to automatically populate a text description of the environment. Overall, the VLM inputs both the RGB image and the text description from θ .

B. From State Descriptions to Decomposed Subtasks

Now that the VLM has the task and state, we move towards outputting DMP weights for robot motion. We divide this overall policy π into two parts: *decomposition* $\pi_{\mathcal{D}}$ and *generation* $\pi_{\mathcal{G}}$. In the decomposition, the VLM outputs a language description of the next subtask τ_i . Each subtask is a step towards the completed behavior. Consider our motivating example of cleaning the plate: this task τ breaks down into three subtasks including grasping the sponge τ_1 , carrying it to the plate τ_2 , and wiping the plate τ_3 . We want each subtask to correspond to something executable by our DMPs. We therefore provide a *template* for the decomposition policy $\pi_{\mathcal{D}}$ to complete at each high-level timestep. The subtasks within this template can be of two forms. First is `ACTION(object)`, where the robot acts on a single object (e.g., “grasp the sponge”). Second is `ACTION(object) TO (object)`, which is used for actions with multiple objects (e.g., “carry the sponge to the plate”). These templates ensure that the decompositions are anchored in subtasks related to scene objects and that subtasks are only involving one primary object that is manipulated, ensuring the right difficulty for each DMP.

With this framework in mind, the decomposition policy $\pi_{\mathcal{D}}$ is a foundational vision and language model of the form:

$$\varphi_i \sim \pi_{\mathcal{D}}(o \mid \tau, o, \theta, [\varphi_k]_0^{i-1}) \quad (7)$$

Here $[\varphi_k]_0^{i-1}$ represents the sequence of previously proposed subtask templates. At the start of the interaction this sequence is empty, and the sequence iteratively grows each time the decomposition policy is queried. Looking at Equation (7), τ is the high level task, and o and θ are the RGB image and natural language state description from Section IV-A. The completed subtask template φ_i output by this model describes the motion that the next DMP should complete.

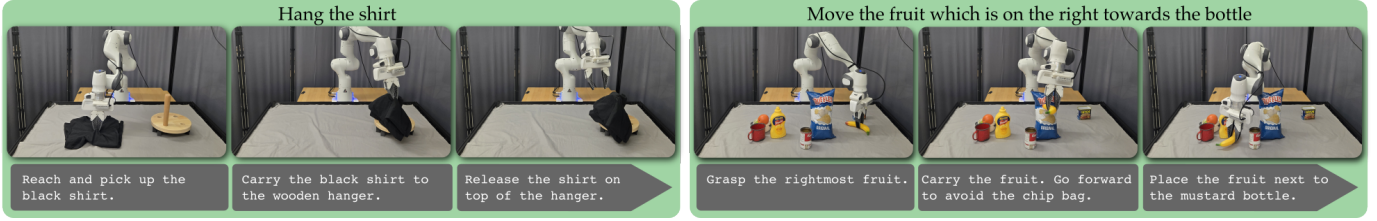


Fig. 3: Our experiments evaluate LMP’s performance on tabletop-manipulation tasks, converting natural-language task descriptions into robot controllers. In our tests, we evaluate 31 diverse household tasks requiring semantic task understanding, awareness of obstacles, and spatial reasoning.

C. From Subtasks to Generating DMPs

Now that we know the next desired motion, we ground this motion in the robot’s physical environment with DMPs. We specifically choose DMPs due to their semantically interpretable parameters: The forcing function \mathbf{W}_i directly shapes the resulting trajectory, allowing for a deterministic change in the robot’s motion along the modeled end-effector path. This property renders DMP weights as a natural action space for a VLM, allowing it to reason over and generate these weights from free-form task descriptions without requiring demonstrations or in-domain fine-tuning. Leveraging this semantic interpretability to ground VLM reasoning in continuous low-level motions is the central contribution of LMP, bridging symbolic task decomposition and embodied trajectory generation.

More specifically, given our scene description and next subtask, we sample from the *generator policy* π_G to produce DMP weights W_i and offset parameters Δz and $\Delta\psi$:

$$(W_i, \Delta z, \Delta\psi) \sim \pi_G(\circ \mid \varphi_i, s_G, o, \theta) \quad (8)$$

The generator VLM produces weights $W_i \in \mathbb{R}^{M \times B}$ where M is the number of controlled degrees of freedom and B denotes the number of basis functions used in the DMP formulation. As a reminder, the basis functions are applied in Equation (3) to shape the DMP trajectory. The details of the grounding prompt s_G will be outlined later in this subsection.

To improve tractability and reduce unnecessary complexity, we restrict the generator to the minimal set of operational dimensions required for task execution. Our experiments focus on top-down robot manipulation tasks where the end-effector is positioned above the target object. We accordingly consider the motion of the robot end-effector in Cartesian space. Under this formulation, the weight matrix W_i is composed of the following weight vectors:

$$\mathbf{w}_i^{(x)}, \mathbf{w}_i^{(y)}, \mathbf{w}_i^{(z)}, \mathbf{w}_i^{(\theta_z)}, \mathbf{w}_i^{(g)} \in \mathbb{R}^B$$

where $\mathbf{w}_i^{(x)}$, $\mathbf{w}_i^{(y)}$, $\mathbf{w}_i^{(z)}$ correspond to translational motion along the Cartesian axes, $\mathbf{w}_i^{(\theta_z)}$ represents the end-effector rotation about the z -axis, and $\mathbf{w}_i^{(g)}$ controls the gripper state. Note that the gripper command is represented as a continuous variable rather than a binary signal, allowing it to be controlled in the same manner as the other degrees of freedom. In practice, we often just need the gripper to open or close at a specific timestep along the trajectory. We encode this knowledge in the basis functions of the gripper DMP by replacing the Gaussian

in Equation (4) with a step function. More generally, designers can modify the DMP hyperparameters (e.g., types and number of basis functions) to capture their domain knowledge.

The weights W_i determine the continuous trajectory *shape*. But we also need to set the *goal* g that the DMP converges towards. This goal is specified in two parts by offsets Δz and $\Delta\psi$. We first extract the location of the center of the object from θ in Section IV-A. We then move towards a point aligned with the object’s xy position, but offset in both height and orientation. The generator policy in Equation (8) outputs a target height along the z -axis (Δz) and a target orientation around the z -axis ($\Delta\psi$).

Overall, the generator π_G provides the parameters that define DMPs in each Cartesian axis. Our experiments show that we can use off-the-shelf vision and language models as π_G , provided that we give these models some context on how DMPs operate. We input this grounding through the task-invariant prompt s_G . This prompt (available in the supplemental material) encourages the generator to internalize the relationship between the physical workspace and the DMP weight space, i.e., how variations in DMP weights induce corresponding changes in Cartesian end-effector motion. For example, the prompt explains that “increasing the weight of parameters in the x -axis will cause the trajectory to move in that direction.” Equipped with s_G , the generator policy is able to translate subtask descriptions into low-level DMP parameters; hence, the subtask φ_i only needs to be in natural, human-understandable language.

After decomposition and generation, we execute the DMP with weights W_i and goal pose g :

$$\mathbf{g} = [p_k^x \quad p_k^y \quad p_k^z + \Delta z \quad p_k^\theta \quad p_k^\phi \quad p_k^\psi + \Delta\psi] \quad (9)$$

Here p_k is the position of the target object. The position of all objects is originally estimated in θ from Section IV-A, and the target object is assigned by subtask φ_i . The robot rolls out this motion in the environment. More precisely, the robot uses the DMP as a reference trajectory, and outputs low-level control commands to track the reference.

D. Using Feedback to Refine the Motion

Ideally, the motion produced in Section IV-C is correct and the robot completes the desired subtask. But in practice this may not be the case; we therefore enable iterative, step-by-step feedback through a *subtask judge*. The subtask judge observes how the environment changes over the course of

TABLE I: Tasks and Performances.

Reported in success rate (%). Numbers in parenthesis (\circ) represent the number of feedbacks per task, averaged across trials. Columns 4-6 refer to our judge-free (J/F), decomposer free (D/F), and judge- & decomposer-free (J/F D/F) ablations. Task Categories: ● Semantic understanding ● Obstacle awareness ● Spatial reasoning

Task Name	●	●	●	LMP	TrajGen [15]	$\pi_{0.5}$ [18]	LMP J/F	LMP D/F	LMP J/F D/F
pick the chip bag on the left of the table			✓	100 (0.2)	40	20	80	40 (2.2)	20
pick the rightmost can			✓	100 (–)	20	40	100	20 (2.4)	20
pick the fruit in the middle			✓	60 (1.2)	60	40	60	80 (0.6)	60
pick the chip bag which is to the right of the can			✓	100 (–)	60	20	100	60 (0.6)	60
move the fruit which is on the right towards the bottle			✓	100 (–)	60	0	100	80 (0.6)	60
move the banana near the pear			✓	100 (–)	100	60	100	80 (0.6)	40
move the banana near the pear (obstacles included)		✓	✓	80 (1.0)	20	60	40	0 (3)	0
move the can to the center of the table			✓	80 (0.6)	20	20	80	80 (0.6)	40
move the lonely object to the others	✓		✓	80 (0.6)	0	60	80	100 (0.2)	80
place the apple in the bowl				60 (1.2)	60	40	60	40 (0.6)	0
place the apple in the bowl (obstacles included)		✓		60 (1.6)	0	20	20	60 (1)	0
pick the apple from the bowl and place it on the table				80 (0.6)	60	40	80	20 (1.2)	20
wipe the plate	✓			80 (0.6)	0	0	80	0 (3)	0
drop the ball into the cup				60 (1.4)	0	20	40	20 (1.2)	0
drop the ball into the cup (obstacles included)		✓		80 (2.4)	0	0	20	20 (1.2)	0
insert the bread into the toaster				80 (0.6)	0	0	80	20 (2.2)	0
pick up the bowl				80 (0.6)	20	40	80	100 (0.4)	60
hang the shirt	✓			40 (1.8)	0	40	40	40 (0.6)	20
put the banana on the plate			✓	80 (0.6)	80	60	80	60 (1.2)	60
put the banana on the plate (obstacles included)		✓	✓	100 (0.4)	0	40	60	40 (3)	0
knock over the left bottle			✓	100 (0.2)	80	60	80	80 (1.6)	60
push the bottle to the orange			✓	40 (1.2)	40	40	40	40 (2.2)	0
push the can towards the right			✓	60 (1.2)	20	0	60	60 (1.2)	60
stir the mug with the spoon	✓			40 (1.6)	0	0	0	20 (2.6)	0
draw a five-pointed star			✓	0 (3)	100	0	0	0 (3)	0
move the pan to the left			✓	60 (1.4)	40	0	40	40 (1.8)	40
wipe the table with the sponge			✓	80 (0.8)	60	60	60	60 (1.8)	40
draw a circle with the gripper closed			✓	0 (3)	100	40	0	0 (3)	0
unplug the charger	✓			20 (2.4)	0	20	20	40 (2.8)	0
take out the tissue from the dispenser	✓			0 (3)	20	20	0	0 (3)	0
open the bottle cap	✓			20 (3)	20	0	0	0 (3)	0
Overall Performance				65	35	28	54	41	24

subtask φ_i , and provides qualitative feedback in text form to the decomposer and the generator. The judge outputs a qualitative feedback statement r that describes the error and/or how the robot should improve (e.g., “too high, you missed the sponge”). To incorporate this feedback and refine the robot’s motion, we add the refinement r to the robot’s existing prompts for both the decomposer and the generator. The environment is then reset to the configuration it was in *before* the DMP was executed, and the decomposer and generator reason about appropriate corrections. Through this in-context learning mechanism, we are able to iteratively improve both the task decomposition and generated weights, leading to safer and more efficient interactions.

V. EXPERIMENTS

To demonstrate the effectiveness of LMP, we compare it to state-of-the-art baselines and conduct extensive ablations in a controlled tabletop manipulation setting involving household objects. We evaluate LMP on 31 household manipulation tasks (Figure 3), covering the full set of tasks evaluated in [15]. We compare overall task success rates against $\pi_{0.5}$ [18] and TrajGen [15]. Furthermore, we ablate our method to

evaluate the impact of free-form natural language feedback on motion generation as well as the impact of grounding subtask identification through task decomposition.

Setup. We evaluate these methods on a 7-DoF Franka Emika Panda robot arm with a UMI gripper (see Figure 3). We use a mounted Orbbec Femto Mega camera for RGB-D imagery. For $\pi_{0.5}$, we instead use two Intel Realsense D435 cameras: one mounted on the robot end-effector and one statically mounted with an isometric view of the workspace, covering the same field of view. We use Gemini Robotics-ER 1.5 [43] to obtain object labels $\{l_i\}_{i=1}^M$. These labels are then passed to LangSAM [53], [54], which produces fine-grained object segmentations. These segmentation masks are subsequently projected onto the depth image to recover the corresponding 3D object bounding boxes. We use these bounding boxes to determine the approximate yaw of each object relative to the end-effector orientation. Leveraging its strong capabilities in physical scene understanding and multimodal reasoning for robotic tasks, we also employ Gemini Robotics-ER as our task decomposer. We use GPT-5.2 [58] as the generator policy. Feedback, when needed, is provided by a single person to ensure consistency across the evaluation.

Tasks. For our experiments, we choose a set of 31 tasks, inspired by [15], which contain a diverse set manipulation challenges including spatial reasoning, semantic scene understanding, and obstacle awareness (see Figure 3).

In contrast to [15], we introduce task variants that require the robot to reason about object interactions and potential collisions along its motion, which are not explicitly mentioned in the task instructions, but are implicitly relevant. To further evaluate our method’s ability to interpret semantically complex tasks, we provide generalized task descriptions. For example, we use instructions such as “Wipe the plate” instead of “Wipe the plate with a sponge.” In these cases, we rely on the decomposer to infer the implicit steps and complete the full sequence of actions.

We expect language-enabled motion generators to exhibit three core capabilities: (1) semantic understanding, inferring intent from under-specified instructions; (2) obstacle awareness, recognizing configurations the robot must avoid; and (3) spatial reasoning, understanding spatial relations between objects in the environment. We label tasks according to these characteristics to analyze method capabilities.

Metrics. We employ two performance metrics across tasks:

- 1) Success rate: For each task, the success rate is evaluated over 5 independent trials. In each trial, the positions and orientations of the target objects, as well as the obstacles and distractors, are randomly initialized. We only consider a task successful if all constituent subtasks are completed successfully and no collisions occur.
- 2) We analyze the failure modes of the model with respect to five common categories.

A. Baseline Comparison

We compare our approach against two representative state-of-the-art baselines that use pretrained language models to guide robot policies from natural language instructions. Our selection of baselines reflects two conventions by which language models are incorporated into robotics: as common sense reasoning agents and language-grounded action generators.

TrajGen [15]. [15] leverages the common sense reasoning capabilities of LLMs to generate executable motion representations. Specifically, TrajGen employs an LLM to synthesize code that generates dense waypoint trajectories based in the task instruction, which are subsequently executed by the robot. To ensure a consistent evaluation, we provide TrajGen with the same information from the object detection pipeline and employ GPT-5.2 as its internal LLM.

Pi-0.5 [18]. Vision-Language-Action models such as $\pi_{0.5}$ leverage pretrained language models to effectively ground language instructions in action and visual perception. This allows VLAs to generalize across tasks by encoding high-level semantic knowledge acquired during large-scale pretraining [6]. We train $\pi_{0.5}$ using an aggregated demonstration dataset across all 31 tasks, with five demonstrations per task. We follow the officially released Libero fine-tuning scheme [18] and fine-tune from the standard checkpoint for 20000 steps.

Baseline Results. We report the performance of LMP when compared to the baselines in Table I. We observe that LMP

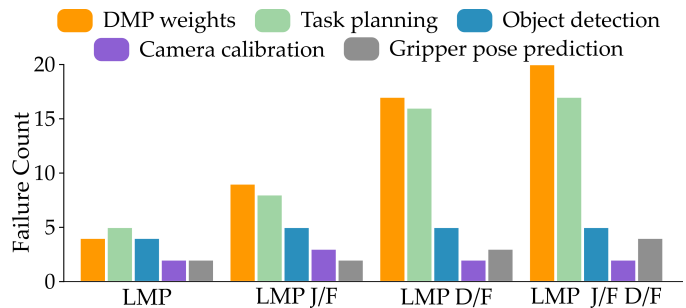


Fig. 4: We identify five dominant failure modes and analyze the impact of the judge as well as the task decomposition on these failure modes across task executions. Particularly the addition of a grounded task-decomposition substantially reduces the errors in task planning and subsequently more effective weight generation. Furthermore, we find that adding feedback has a strong influence on successful DMP weight generation.

consistently outperforms TrajGen and $\pi_{0.5}$ (col. 1-3) with an overall success rate of 65% as compared to 35% and 28% for TrajGen and $\pi_{0.5}$ respectively. We attribute this improvement to decoupling high-level semantic reasoning from low-level motion generation: the semantically interpretable parameter space allows general-purpose foundation models to specify task intent without requiring knowledge of robot dynamics, while the low-level controller handles trajectory execution.

We observe that TrajGen particularly struggles with tasks that require nonlinear motions, such as “wipe the plate”, which requires the robot to move in a circular pattern. TrajGen tends to output linear motion, whereas LMP is capable of generating smooth splines. Additionally, in tasks that require semantic understanding, TrajGen often neglects task-relevant objects that are not directly specified in the task description.

We also found that $\pi_{0.5}$ experiences causal confusion [59]. For example, if an orange is always near an apple during data collection, $\pi_{0.5}$ will fail to develop a strong semantic distinction between these two objects. We also observe that when the task-relevant entities do not provide a strong visual signal, such as referencing the “middle of the table” as opposed to “the plate” as a target position, it is more challenging for $\pi_{0.5}$ to complete the task. This observation is common when fine-tuning a vision-language backbone for low-level control, as it diminishes its overall reasoning capability [60], [61].

B. Ablation Study

Two key components of our method are the ability to incorporate feedback and the utilization of a task decomposition module. To assess the impact of these two components, we conducted extensive ablation studies (see Table I, col 4-6).

Judge-Free Ablation (J/F). We test our method without the judge feedback outlined in Section IV-D. This ablation (LMP J/F) focuses the success of generating DMP weights on the first try, removing the ability to refine the motion through feedback.

Decomposer-Free Ablation (D/F). We also evaluate our method without the task decomposer described in Section IV-B

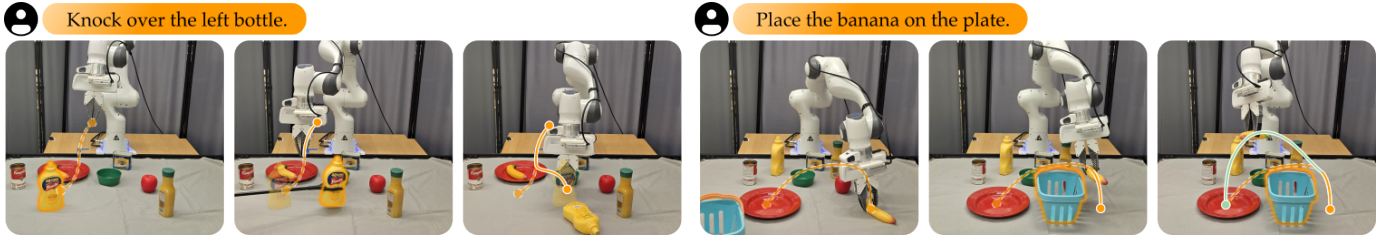


Fig. 5: We evaluate LMP in dynamic environments where object positions and scene configurations may change during execution. *Left*: The target object is moved during execution. In this case, LMP does not need to regenerate DMP weights, and instead adapts by updating the target goal through the goal-attractor dynamics of the DMP. *Right*: New obstacles are introduced along the robot’s path. In this scenario, LMP updates the DMP weights in closed loop to reshape the trajectory and avoid the obstacle.

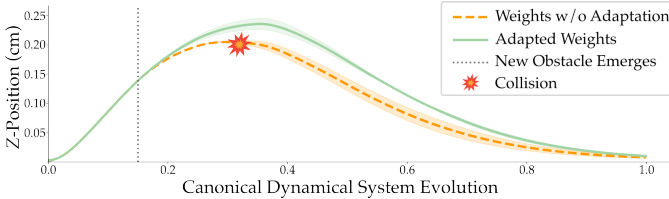


Fig. 6: LMP with support for real-time weight adaptation is able to reactively adjust to changes in the environment during rollout, such as the introduction of new obstacles, to avoid collision while still converging to the goal position and completing the task.

(LMP D/F). In this setting, the generator VLM is prompted to infer a sequence of DMPs, goals, and offsets, decomposing the task implicitly. This ablation aims to demonstrate the importance of a dedicated decomposer to anchor the DMP generation in appropriate subtask complexity.

Judge- & Decomposer-Free Ablation (J/F D/F). We evaluate a further ablation (LMP J/F D/F) of *Decomposer-Free* by removing expert feedback from the generator policy. This ablation intends to show that expert feedback is especially critical when the generator lacks structured subtask grounding.

Ablation Results. We find that removing either the judge or the decomposer leads to reduced performance. J/F achieves a success rate of 54%, while removing the decomposer D/F degrades performance to 41%. Removing both results in a success rate of 24%. The sources of failures differs across ablations, as shown in Figure 4. We find that both the judge and decomposer are necessary components to improve *DMP Weight Generation* and *Task Planning*. Removing the decomposer leads to a major degradation in overall task performance. Without it, LMP tends to manipulate task-irrelevant objects and reason about the task at an inappropriate level of abstraction: either too complex to be effectively captured by DMP parameterization, or too granular, at which point general-purpose foundation models lack the required knowledge to successfully reason over embodied robot dynamics. Furthermore, the judge particularly improves weight generation by allowing LMP to refine DMP parameters.

In summary, these ablations highlight the complementary roles of the judge and decomposer in grounding the foundation

model at an appropriate level of abstraction.

C. Reactive Adaptation

Reactive adaptation to changing environments is crucial in task and motion planning. Object positions and scene configurations may evolve during execution, and robotic systems should be able to adapt to these changes in real-time. To this end, we evaluate a version of LMP with support for real-time adaptation (LMP-Adapt). This version demonstrates that LMs are able to reason over the weights of DMPs when shifting targets and changes in obstacle configurations occur. Specifically, we compare LMP-Adapt against baseline methods on two categories of dynamic tasks (Table II).

Moving target objects. In these tasks, we evaluate scenarios with dynamically changing goal positions g while preserving the overall motion structure generated by the VLM. We use a video segmentation and tracking model [62] to track the objects involved in the current subtask and update the DMP goal position online during execution. This setting is designed to demonstrate that the goal-attractor dynamics of DMPs lends itself well to dynamic environments.

Dynamic obstacles. In these tasks, a VLM continuously incorporates new observations during rollouts and updates the DMP weights in a closed loop manner, enabling responsive replanning as the environment evolves. Importantly, while the goal position remains the same, this task evaluates whether or not the shape of the motion itself adapts to obstacles that need to be avoided during the execution of the motion. DMPs lend themselves particularly well to this setting as their goal-attractor dynamics continue to converge to the goal, while the weights of the forcing terms allow the VLM to re-shape the motion itself in the semantically interpretable parameter space to avoid the obstacle. We consider two scenarios that alter the trajectory shape: the introduction of a new obstacle along the robot’s path, and the removal of an existing obstacle during execution. Specifically, we use GPT-5.4-nano [63] as a lightweight VLM that provides interpretable outputs while supporting closed-loop updates at a reasonable frequency of approximately 0.3 Hz. This setting is designed to demonstrate that, despite being bottlenecked by language model inference time, LMP with weight adaptation is still capable of performing closed-loop control and producing semantically

TABLE II: Reactive Planning Performances. Reported in success rate (%). **Bolded objects** are being moved during the rollouts.

Moving Target Objects	LMP	TrajGen [15]	$\pi_{0.5}$ [18]
place the apple in the bow l	60	0	20
wipe the plate	60	0	20
hang the shirt (hanger)	20	0	0
knock over the left bottle	80	0	0
stir the mug with the spoon	20	0	0
Overall Performance	48	0	8

meaningful and reasonable trajectory adjustments in response to environmental changes.

Our results in Table II and Table III show that LMP outperforms the baselines across both categories of dynamic tasks. In the moving target object scenarios, TrajGen is unable to adjust its trajectory because its execution is fully open loop. In contrast, while $\pi_{0.5}$ demonstrates certain ability to track moving objects, we observe that it is more prone to going out of distribution under visual disturbances and changing scene configurations, causing an overall performance drop of 20%. In the dynamic obstacle scenarios, we observe that LMP-Adapt is sometimes unable to adjust its trajectory quickly enough in response to environmental changes due to the limited update frequency of the VLM (recall that we are using 0.3 Hz for the live updates). Nevertheless, across the two tasks we evaluate, LMP-Adapt does not show a significant drop in success rate compared to the static scene setting, retaining a performance of 55%, whereas $\pi_{0.5}$ show noticeable performance degradation to 5%. We show LMP bending the trajectory shape along the z -axis in response to emerging obstacles across rollouts in Figure 6.

From the per-DoF changes presented in Table III, we observe that the weight modifications induced by the language model in response to dynamic objects primarily affect the z direction, which is expected as the robot mainly moves in the z -axis to avoid objects. We also observe changes along the y -axis. However, in contrast to the z -axis, the motion along the y axis mainly influences the temporal progression of the motion to allow for the spatial adjustment of the z -axis. We test this hypothesis by applying Dynamic Time Warping (DTW) [64] in addition to plainly computing the distance between corresponding trajectory points. We observe that the accumulated deviation along y over 1000 timesteps remains relatively small (9.238 ± 7.700 cm) compared to the much larger changes observed along z (92.06 ± 156.8 cm), supporting our observation that changes in y are largely temporal, not spatial adjustments. In particular, when obstacle configurations change, the adapted y -axis weights enable the robot to slow down and temporally coordinate its motion with the concurrent height adjustments along the z -axis required for obstacle avoidance. These results suggest that the language model is capable of reasoning over spatial relationships and expressing such reasoning through meaningful adaptations of the DMP trajectory weights.

TABLE III: Dynamic obstacle performance and per-DoF changes. Performance is reported as success rate (%); Per-DoF changes (cm) over 1000 timesteps, computed as the accumulated absolute timestep-wise difference between trajectories generated from the original and adapted weights.

Dynamic Obstacles	LMP-Adapt	LMP	TrajGen [15]	$\pi_{0.5}$ [18]
T1. put the banana on the plate (emerging obstacle)	40	0	40	20
T2. put the banana on the plate (obstacle removed)	100	80	80	0
T3. drop the ball into the cup (emerging obstacle)	20	0	0	0
T4. drop the ball into the cup (obstacle removed)	60	60	0	0
Overall Performance	55	35	30	5

	x	y	z	yaw
T1.	117.7 \pm 235.5	409.5 \pm 358.1	790.9\pm587.7	0.000 \pm 0.000
T2.	94.03 \pm 188.1	473.6 \pm 116.1	1092\pm22.54	0.000 \pm 0.000
T3.	40.97 \pm 2.284	551.3 \pm 25.80	1098\pm9.824	1.236 \pm 0.07068
T4.	68.39 \pm 136.8	615.1 \pm 2.630	1080\pm0.001	4.202 \pm 7.352

TABLE IV: CALVIN experiments; 5 task chains
Tasks: open drawer; slide door left/right; place cube in drawer, close drawer, turn on/off led

$\varnothing \rightarrow D$ | S1: 100% | S2: 85.7% | S3: 71.4% | S4: 42.8% | S5: 42.8%

D. Long-Horizon Benchmark

To show that LMP can effectively handle long-horizon manipulation tasks, we evaluate it on a subset of the CALVIN tasks [65]. Specifically, compared to the original CALVIN task, we use a modified version that consisting of a preliminary subset of five representative manipulation tasks from the benchmark, including pick-and-place, pushing, and pulling behaviors. LMP achieves promising sequential success rates across five randomly ordered CALVIN tasks (Table IV), demonstrating easy adaptation to novel benchmarks beyond pick-and-place tasks while using the same task-invariant grounding prompt. Particularly for the long-horizon case with a sequence of five subtasks, LMP is capable of chaining and generating motions from the result of prior executions, resulting in an overall zero-shot success rate of 42.8%. These results further highlight the scalability of LMP, showing that the framework can generalize to more diverse manipulation settings without task-specific redesign.

E. Other Action Representations

We use DMPs to model the end effector’s position and orientation in Euler-angle space throughout our evaluations. However, the core insight of VLM-based controller generation is not limited to DMPs. Fundamentally, the key insight in LMP is that semantically interpretable parameters provide a capable interface to connect LLM and VLM based reasoning with embodied low-level control. To this end, the following demonstrates adaptations to different action spaces as well as a discussion of alternative controllers.

In terms of action parameterization, besides using Euler angles for rotation representation, we also explore quaternions as an alternative expressive rotational parameterization. Across 11 of the *spatial reasoning* tasks, chosen as a subset from Table I for their need for rotational alignment, LMP maintains strong performance under both representations, achieving overall success rates of 82% and 89% with quaternion

and Euler-angle parameterizations respectively. These results demonstrate the adaptability of LMP to different action representations.

In terms of controller formulation, LMP currently focuses on point-attractor DMPs, while extensions incorporating alternative controllers are left for future work. Other trajectory generation methods, such as B-spline parameterizations [66], could also be integrated within the same framework. We choose DMPs over alternative trajectory representations for several reasons. A key motivation is that many manipulation tasks, such as top-down grasping, wiping, and obstacle avoidance, fundamentally require shaped motions, where directly moving toward a goal attractor would be infeasible. DMPs naturally support such structured trajectories through their learned forcing functions, while still maintaining stable goal-directed behavior. In addition, DMP weights carry temporally grounded semantics, making them particularly amenable to reasoning and generation by VLMs.

Another important advantage is that the goal-attractor dynamics of DMPs enable online adaptation to moving targets through simple updates to the goal state g , retaining the overall motion structure generated by the VLM. Such reactive adaptation is less straightforward to achieve with geometric trajectory representations such as B-splines. In practice, the generated DMP is executed by sampling the trajectory at a fixed frequency and leveraging the robot’s position-control API. Furthermore, the closed-loop nature of DMPs [26], [52] provides temporal corrections through error calculation and stable convergence behavior, as the influence of the forcing function gradually decays to zero over the course of execution.

VI. CONCLUSION

We presented a framework to bridge the gap between language understanding and embodied motion control. Our core contribution is a novel framework to ground the common-sense reasoning capabilities of vision-language models into embodied motion through control-theoretic approaches leveraging dynamic movement primitives. We argue that this combination is effective because DMPs have a relatively small number of semantically meaningful parameters that VLMs can inherently tune. Through extensive experiments in static and dynamic environments, we demonstrate that the VLM is cable to specify the necessary weights, thereby translating high-level reasoning into low-level embodied behaviors. We evaluate our approach through 31 tabletop manipulation tasks: when using LMP, the robot completed tasks on its first attempt 54% of the time, and 65% of the time when given up to 3 rounds of natural language corrections. This contrasts with VLM approaches that do not incorporate DMPs (35% success rate), and with robotics foundation models that need in-domain fine-tuning (28% success rate).

Limitations and Future Work. While LMP provides a framework that bridges the gap between high-level semantic reasoning and low-level control, it requires the low-level controller to have a set of semantically interpretable parameters. While this work focuses on the use of DMPs, we hypothesize that the proposed framework can be extended in future work

to incorporate other classes of controllers as long as their respective parameters are semantically interpretable for a VLM. Furthermore, live updates in dynamic environments are limited by how fast a VLM can respond (currently $\approx 0.3\text{Hz}$). Future work needs to be done to assess the framework’s ability to reason over relative spatial relationships. In particular, we observe that the language model fails more frequently when grounding trajectory weights from abstract relative instructions such as “10 cm to the left”. Moreover, our evaluation is currently limited to five degrees of freedom. Extending the framework to higher-dimensional action spaces would introduce additional challenges for language models to reason over the motion, but represents an important direction for future investigation with will also scale with further advances in VLMs themselves. Finally, while feedback from a human judge has proven useful, we will investigate the use of an autonomous judge, such as a VLM, in future work.

REFERENCES

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.
- [2] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine, “The ingredients for robotic diffusion transformers,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 15617–15625.
- [3] O. M. Team *et al.*, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024.
- [4] D. Kalashnikov *et al.*, “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning (CoRL)*, 2018.
- [5] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4238–4245.
- [6] B. Zitkovich, T. Yu, S. Xu, P. Xu *et al.*, “RT-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning*, 2023, pp. 2165–2183.
- [7] M. J. Kim, K. Pertsch, S. Karamcheti *et al.*, “OpenVLA: An open-source vision-language-action model,” in *Conference on Robot Learning (CoRL)*, 2024.
- [8] K. Black, N. Brown, D. Driess, A. Esmail *et al.*, “ π_0 : A vision-language-action flow model for general robot control,” *arXiv preprint arXiv:2410.24164*, 2024.
- [9] S. Ye, J. Jang, B. Jeon, S. Joo *et al.*, “Latent action pretraining from videos,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [10] Q. Li, Y. Deng, Y. Liang, L. Luo *et al.*, “Scalable vision-language-action model pretraining for robotic manipulation with real-life human activity videos,” *arXiv preprint arXiv:2510.21571*, 2025.
- [11] P. Intelligence *et al.*, “ $\pi_{0.6}^*$: a VLA that learns from experience,” *arXiv preprint arXiv:2511.14759*, 2025.
- [12] A. Velasquez, N. Bhatt, U. Topcu, Z. Wang, K. Sycara, S. Stepputtis, S. Neema, and G. Vallabha, “Neurosymbolic ai as an antithesis to scaling laws,” *PNAS Nexus*, vol. 4, no. 5, p. pgaf117, 2025.
- [13] M. Ahn, A. Brohan, N. Brown, Y. Chebotar *et al.*, “Do as I can and not as I say: Grounding language in robotic affordances,” in *Conference on Robot Learning (CoRL)*, 2023.
- [14] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” *arXiv preprint arXiv:2209.07753*, 2022.
- [15] T. Kwon, N. Di Palo, and E. Johns, “Language models as zero-shot trajectory generators,” *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6728–6735, 2024.
- [16] M. T. Khan and A. Waheed, “Foundation model driven robotics: A comprehensive review,” *arXiv preprint arXiv:2507.10087*, 2025.
- [17] S. Bai, W. Song, J. Chen, Y. Ji *et al.*, “Embodied robot manipulation in the era of foundation models: Planning and learning perspectives,” *arXiv preprint arXiv:2512.22983*, 2025.

- [18] P. Intelligence *et al.*, “ $\pi_{0.5}$: a vision-language-action model with open-world generalization,” in *Conference on Robot Learning (CoRL)*, 2025.
- [19] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar *et al.*, “RT-1: Robotics transformer for real-world control at scale,” in *Robotics: Science and Systems (RSS)*, 2023.
- [20] C. Lynch and P. Sermanet, “Language conditioned imitation learning over unstructured data,” in *Robotics: Science and Systems (RSS)*, 2020.
- [21] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, “Language-conditioned imitation learning for robot manipulation tasks,” *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 13 139–13 150, 2020.
- [22] K. Kawaharazuka, J. Oh, J. Yamada, I. Posner, and Y. Zhu, “Vision-language-action models for robotics: A review towards real-world applications,” *IEEE Access*, 2025.
- [23] S. Bhagat, S. Li, J. Campbell, Y. Xie, K. Sycara, and S. Stepputtis, “Let me help you! Neuro-symbolic short-context action anticipation,” *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9749–9756, 2024.
- [24] L. Keller, D. Tanneberg, and J. Peters, “Neuro-symbolic imitation learning: Discovering symbolic abstractions for skill learning,” *arXiv preprint arXiv:2503.21406*, 2025.
- [25] J. Campbell and H. B. Amor, “Bayesian interaction primitives: A SLAM approach to human-robot interaction,” in *Conference on Robot Learning*, 2017, pp. 379–387.
- [26] S. Schaal, “Dynamic movement primitives—a framework for motor control in humans and humanoid robotics,” *Adaptive Motion of Animals and Machines*, pp. 261–280, 2006.
- [27] N. M. Shafiqullah, Z. Cui, A. A. Altanzaya, and L. Pinto, “Behavior transformers: Cloning k modes with one stone,” *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 22 955–22 968, 2022.
- [28] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, “Deep imitation learning for bimanual robotic manipulation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 2327–2337.
- [29] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5628–5635.
- [30] T. Haarnoja, B. Moran, G. Lever, S. H. Huang *et al.*, “Learning agile soccer skills for a bipedal robot with deep reinforcement learning,” *Science Robotics*, vol. 9, no. 89, p. eadi8022, 2024.
- [31] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, “How to train your robot with deep reinforcement learning: Lessons we have learned,” *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.
- [32] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.
- [33] J. Luo, Z. Hu, C. Xu, Y. L. Tan *et al.*, “SERL: A software suite for sample-efficient robotic reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 961–16 969.
- [34] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.
- [35] E. Elelimy, D. Szepesvari, M. White, and M. Bowling, “Rethinking the foundations for continual reinforcement learning,” *arXiv preprint arXiv:2504.08161*, 2025.
- [36] J. Hu, Z. Lian, Z. Wen, C. Li, G. Chen, X. Wen, B. Xiao, and M. Tan, “Continual knowledge adaptation for reinforcement learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- [37] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, “Mastering diverse domains through world models,” *arXiv preprint arXiv:2301.04104*, 2024.
- [38] G. Zhou, H. Pan, Y. LeCun, and L. Pinto, “DINO-WM: World models on pre-trained visual features enable zero-shot planning,” in *International Conference on Machine Learning (ICML)*, 2024.
- [39] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna *et al.*, “DROID: A large-scale in-the-wild robot manipulation dataset,” in *Robotics: Science and Systems*, 2024.
- [40] Y. Dai, A. Keyser, and D. P. Losey, “Prepare before you act: Learning from humans to rearrange initial states,” *arXiv preprint arXiv:2509.18043*, 2025.
- [41] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta *et al.*, “Open X-embodiment: Robotic learning datasets and RT-X models: Open X-embodiment collaboration⁹,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.
- [42] M. Dalal, M. Liu, W. Talbott, C. Chen, D. Pathak, J. Zhang, and R. Salakhutdinov, “Local policies enable zero-shot long-horizon manipulation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 13 875–13 882.
- [43] G. R. Team *et al.*, “Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer,” *arXiv preprint arXiv:2510.03342*, 2025.
- [44] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, “Vision-language models are zero-shot reward models for reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [45] Y. Liang, N. Kumar, H. Tang, T. Silver *et al.*, “VisualPredicator: Learning abstract world models with neuro-symbolic predicates for robot planning,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [46] A. Capitanelli and F. Mastrogiovanni, “A framework for neurosymbolic robot action planning using large language models,” *Frontiers in Neurorobotics*, vol. 18, p. 1342786, 2024.
- [47] S. Li, S. Bhagat, J. Campbell, Y. Xie, W. Kim, K. Sycara, and S. Stepputtis, “ShapeGrasp: Zero-shot task-oriented grasping with large language models through geometric decomposition,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 10 527–10 534.
- [48] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, “Dynamic movement primitives in robotics: A tutorial survey,” *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.
- [49] S. Li, W. Huang, C. Miao, K. Xu, Y. Chen, T. Sun, and Y. Cui, “Efficient robot manipulation via reinforcement learning with dynamic movement primitives-based policy,” *Applied Sciences*, vol. 14, no. 22, 2024.
- [50] H. Zhou, M. Ding, W. Peng, M. Tomizuka, L. Shao, and C. Gan, “Generalizable long-horizon manipulations with large language models,” *arXiv preprint arXiv:2310.02264*, 2023.
- [51] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, “Physically grounded vision-language models for robotic manipulation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 12 462–12 469.
- [52] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, “ProDMP: A unified perspective on dynamic and probabilistic movement primitives,” *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2325–2332, 2023.
- [53] N. Ravi, V. Gabeur, Y.-T. Hu *et al.*, “SAM 2: Segment anything in images and videos,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [54] S. Liu, Z. Zeng, T. Ren *et al.*, “Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection,” in *European Conference on Computer Vision (ECCV)*, 2024, pp. 38–55.
- [55] R. Khanam and M. Hussain, “YOLOv11: An overview of the key architectural enhancements,” *arXiv preprint arXiv:2410.17725*, 2024.
- [56] M. Minderer, A. Gritsenko, A. Stone *et al.*, “Simple open-vocabulary object detection with vision transformers,” in *European Conference on Computer Vision (ECCV)*, 2022, pp. 728–755.
- [57] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, “Open-vocabulary object detection via vision and language knowledge distillation,” in *International Conference on Learning Representations (ICLR)*, 2021.
- [58] A. Singh, A. Fry, A. Perelman, A. Tart *et al.*, “OpenAI GPT-5 system card,” *arXiv preprint arXiv:2601.03267*, 2025.
- [59] Y. Dai, R. R. Sanchez, R. Jeronimus, S. Sagheb, C. M. Nunez, H. Nemlekar, and D. P. Losey, “Civil: Causal and intuitive visual imitation learning,” *arXiv preprint arXiv:2504.17959*, 2025.
- [60] Z. Zhou, Y. Zhu, M. Zhu, J. Wen, N. Liu, Z. Xu, W. Meng, Y. Peng, C. Shen, F. Feng *et al.*, “Chatvla: Unified multimodal understanding and robot control with vision-language-action model,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 5377–5395.
- [61] S. Dey, J.-N. Zaech, N. Nikolov, L. Van Gool, and D. P. Paudel, “Revla: Reverting visual domain limitation of robotic foundation models,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 8679–8686.
- [62] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee, “Tracking anything with decoupled video segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1316–1326.
- [63] OpenAI, “Introducing gpt-5.4 mini and nano,” <https://openai.com/index/introducing-gpt-5-4-mini-and-nano/>, 2026, accessed: 2026-05-11.

- [64] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.
- [65] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.
- [66] H. Zhou, W. Liao, X. Huang, Y. Tang, F. Otto, X. Jia, X. Jiang, S. Hilber, G. Li, Q. Wang *et al.*, "Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning," *Advances in Neural Information Processing Systems*, vol. 38, pp. 172 934–172 959, 2026.