IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Enhancing Robot Learning Through Cognitive Reasoning Trajectory Optimization Under Unknown Dynamics

Qingwei Dong, Tingting Wu, Peng Zeng*, Chuanzhi zang, Guangxi Wan*, Shijie Cui

***Abstract*—In the domain of robot learning, equipping robots with the capability to swiftly acquire operational skills poses a significant challenge. Currently, reinforcement learning techniques are adept at addressing dynamic, unstructured problems involving rich contact scenarios. However, the convergence rate of these algorithms is often slow due to the high dimensionality of the robot state-action mapping space and the extensive initial policy search space. Meanwhile, advancements in large language models (LLMs) have endowed these models with a degree of logical reasoning ability, enabling them to take goal-oriented actions proactively during the initial phase of a robotic task. These models can implicitly generate features of states and uncover underlying patterns in trajectory generation. Yet, in complex manipulative tasks involving rich contact scenarios, LLMs still fall short. Thus, integrating the robust interactive capabilities of reinforcement learning with the strong logical reasoning of LLMs, and enhancing policy search with LLMs, could potentially accelerate the speed of policy searches significantly. In this paper, we introduce a Cognitive Reasoning Trajectory Optimization method. This approach utilizes Low-level Cognitive Control Tuning to enable LLMs with robust logical reasoning to make effective single-step decisions in Markov Decision Process (MDP) tasks. By fitting dynamic models with high-quality cognitive reasoning data and optimizing control strategies, this method constrains the policy search space and enhances the efficiency of trajectory optimization. Experimental results on various manipulative tasks using the Sawyer robot in the Mujoco simulator validate the effectiveness of the proposed algorithm.**

***Index Terms*—Trajectory Optimization; Reinforcement Learning; Policy Search; Large Language Models.**

I. Introduction

In recent years, reinforcement learning has achieved unprecedented success in robot learning. Typically, researchers model a robotic manipulation task as a MDP and then seek an approximately optimal control strategy through policy search. However, due to challenges such as the high dimensionality of the robot state-action mapping space and slow policy search speeds, various model-based reinforcement learning methods have been proposed [1]. Represented by the Guided Policy Search (GPS) [2] algorithm, model-based reinforcement learning methods have demonstrated convergence speeds up to two orders of magnitude faster than model-free approaches [3]. However, accelerating policy search remains a relentless pursuit in the robot academic community [1].

Currently, LLMs demonstrate exceptional logical reasoning capabilities across a diverse array of task planning scenarios, as evidenced by methods such as VoxPoser [4], SayCan [5], HuggingGPT [6], Chain-of-Thought [7], Code as Policies [8], and Inner Monologue [9]. These methods explore LLMs controls of robots from the perspectives of planning, reasoning, and failure handling. Yet, these studies are primarily focused on high-level task planning, leveraging LLMs's fuzzy reasoning to orchestrate sub-strategies. As LLMs continue to evolve, it is inevitable that future LLMs strategies will achieve direct control over lower-level robot operations. Compared to the extensive task planning capabilities of LLMs, the underlying control of robots remains the primary bottleneck. For instance, even if a robot understands the steps required to prepare various breakfasts, simpler tasks such as opening a refrigerator pose greater challenges. Consequently, the question arises whether this advanced reasoning ability could directly assist in policy search within low-level control.

When tasks are appropriately defined with clear specifications and objectives, LLMs with certain logical reasoning capabilities can proactively engage in goal-oriented actions [10]. Additionally, with minimal sample fine-tuning, these models can implicitly generate state characteristics and uncover underlying patterns in trajectory generation. While existing LLMs still fall short in reasoning capabilities for complex operational tasks, they have demonstrated their potential in simpler scenarios. For example, during the initial phase of a robotic manipulation task, LLMs can guide an agent to make significant progress towards a goal. It is well known that during the early stages of reinforcement learning, an agent performs policy search in a vast space, resulting in slow policy convergence. Utilizing LLMs to assist in policy search could substantially accelerate the speed of policy convergence by providing direction in the initial search phase.

In this paper, we successfully incorporate LLMs with significant logical reasoning capabilities into trajectory optimization

Manuscript received: September 5, 2024; Revised: November 2, 2024; Accepted: March 25, 2025. This paper was recommended for publication by Editor Tetsuya Ogata upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China 92267205, 92067205, 92367301 and 92267301, the Natural Science Foundation of Liaoning Province 2024-MSBA-83, the State Key Laboratory of Robotics of China 2023-Z15 and the National Program for Funded Postdoctoral Researchers GZB20230805. *(Corresponding author: Peng Zeng and Guangxi Wan)*

Qingwei Dong, Peng Zeng, Guangxi Wan and Shijie Cui is with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences; Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, ShenYang 110016, China; Qingwei Dong is also with the University of Chinese Academy of Sciences, Beijing 100049, China. Tingting Wu is with the China Mobile Research Institute, Beijing 100053, China. Tingting Wu and Qingwei Dong contributed equally to this paper. Chuanzhi zang is with the School of Artifcial Intelligence, Shenyang University of Technology, Liaoning, Shenyang 110870, China.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

algorithms to guide the direction of the search. Initially, we apply LLMs to low-level robotic control by fine-tuning with a minimal set of incomplete trajectory samples, exceptional handling, and memory modules to achieve single-step decision-making in MDP tasks. Subsequently, we fit dynamic models and optimize control strategies based on historical samples generated by LLMs, thereby constraining the policy search space and enhancing the efficiency of trajectory optimization.

The main contributions of this work are as follows:

(1) We introduce a technique for fine-tuning LLMs, termed Low-level Cognitive Control Tuning (LCCT), validating the feasibility of using LLMs for low-level MDP control tasks in robotics; (2) We propose a method, Cognitive Reasoning Trajectory Optimization (CRTO) under unknown dynamics, enhanced by LLMs. This approach leverages the strong logical reasoning capabilities of LLMs to guide the direction of robot policy search, thereby accelerating the policy search process; (3) Extensive experimental validation. We conducted experiments on three complex manipulation tasks within continuous world environment, and the results confirm the effectiveness of the proposed algorithms.

II. RELATED WORK

Policy search algorithms typically involve executing an initial policy, interacting with the environment, learning a dynamic model, and updating the policy accordingly. Most existing methods employ task-specific priors, including those for dynamic systems [11] [12], policy parameters and structures [13] [14] [15], expected returns [16] [17] [18], and the learning of dynamic system models [19] [20], to endow robots with innate foundational knowledge. While such priors significantly speed up the policy search process, they have a major limitation: experts must provide them for every possible task the robot might face. The emergence of LLMs with strong reasoning abilities offers potential solutions for these broader prior needs. Current research in robotic control using LLM-based methods mainly falls into three categories: reasoning, planning, and control. These correspond to addressing challenges in cognitive decision-making, high-level planning, and policy learning, respectively.

In reasoning, structuring the problem-solving process into sequential or hierarchical steps has facilitated the generation of clearer reasoning pathways by designing mechanisms that enable LLM outputs to reflect sequences of thought or action [21]. Prominent methodologies include the Chain of Thought (CoT) [7], Tree of Thought (ToT) [22], and Graph of Thought (GoT) [23]. CoT uses 'think step by step' prompts for mathematical reasoning; ToT evaluates multiple reasoning paths with value feedback and global selection; GoT models LLM-generated information as graphs, boosting capabilities through networked reasoning. Currently, these approaches, through complex prompting strategies, address sophisticated tasks but are only deployed in lower dimensions of reasoning and have not yet proven effective in the complex, high-dimensional spaces of continuous robotic control.

In planning, researchers utilize high-level strategies to perform complex robotic tasks with long-term horizons. The paper [24] introduces the Plan-Seq-Learn (PSL) method, using LLMs for high-level planning in long-horizon robotic tasks. Study [25] explores LLMs as universal planning generators by creating efficient Python code for tasks in a PDDL domain, though it's suited mainly for manually designed generic plans. SayCan [5] uses LLMs in dialogue to select actions based on reinforcement-learned value functions. Paper [26] links symbolic task planning with continuous operations, using pretrained LLMs to propose symbolic sequences and select action parameters for motion planning. VoxPoser [4] employs LLMs to generate 3D Value Maps and dense end-effector path sequences for manipulation tasks. Paper [27] uses LLMs' summarization to learn user preferences from prior interactions. However, these methods inherently do not allow LLMs to query or intervene in environments, limiting their capacity for intentional information aggregation.

In policy learning, this approach employs sequence modeling to directly predict actions, thereby generating policies. GraspGPT [28] utilizes the open-ended semantic knowledge from LLMs to achieve zero-shot generalization for task-oriented grasping beyond the training dataset. Code as Policies [8] parametrizes low-dimensional inputs of control primitives through hierarchical generation of policy code, facilitating robotic control. Paper [29] employs object detection and segmentation visual models to directly predict dense sequences of end-effector poses in manipulation tasks. Paper [30] extracts internal knowledge about robotic motion from LLMs to reduce the complexity of samples in robotic manipulation. Although these methods have successfully executed various robotic manipulations, directly employing LLMs as policy networks demands substantial computational resources and extensive fine-tuning with expert data for specific tasks.

Our approach aims to enhance the policy search process at a lower control hierarchy through inferential reasoning. By decomposing the policy learning process into cognitive reasoning and trajectory optimization under unknown dynamics, our method leverages the implicit features of state-action sequences via LLMs, while capitalizing on the strengths of reinforcement learning techniques in unstructured environments. Initially, LLMs address the cognitive reasoning segment, focusing on the extraction and understanding of deep features and patterns within state-action sequences. This facilitates the construction of a robust state-space representation, supporting subsequent decision-making processes. During the trajectory optimization phase, faced with the challenge of unknown dynamics, reinforcement learning techniques are employed to determine the optimal policy through trial-and-error learning. This phase particularly emphasizes learning dynamic models and optimizing trajectories to meet predetermined performance standards. The innovation of our framework lies in its integration of the comprehension capabilities of language models with the adaptability of reinforcement learning to address complex decision-making challenges. This methodology not only improves the performance of the policy but also significantly reduces the amount of training data required, thereby enhancing the feasibility and efficacy of the model in efficient robotic skill acquisition.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

III. PRELIMINARIES

Robots represent a typical example of intelligent agents with high-dimensional state spaces. In this paper, we model continuous robotic operations as a discrete MDP. We define the robot state at time step t as x_t, consisting of joint angles (7 dimensions) and joint velocities (7 dimensions). The robot action at time step t, u_t (4 dimensions), includes changes in the Cartesian space of the end effector and increments in the gripper engagement. The trajectory generated by the MDP is defined as $\tau = \{x_1, u_1, \ldots, x_t, u_t, \ldots, x_T, u_T\}$, where $T \in \mathbb{N}$, with $\mathbb{N}$ representing the positive integers.

The system dynamics model is defined as a time-varying linear Gaussian distribution $p(x_{t+1}|x_t, u_t) = N(f_{xt}x_t + f_{ut}u_t + f_{ct}, F_t)$, with f_{x_t}, f_{u_t}, and f_{c_t} as the mean parameters of the dynamics model, and F_t describing the covariance. The control policy to be learned is defined as $p_\theta(u_t \mid x_t) = \mathcal{N}(K_t x_t + k_t, C_t)$, where K_t and k_t describe the linear relationships of the policy, C_t represents the covariance, and θ denotes the formal parameters of the policy. Hence, the MDP trajectory can be described by the distribution: $p(\tau) = p(x_0)\prod_{t=1}^{T} p_\theta(u_t|x_t)p(x_{t+1}|x_t, u_t)$. The initial state distribution of the system is denoted as $p(x_0)$.

In this trajectory optimization problem, the objective is to learn a policy p_θ that minimizes the expected loss over the entire trajectory:

$$\theta^* = minE_{p_\theta} \sum_{t=1}^{T} l(x_t, u_t), T \in \mathbb{N} \tag{1}$$

where $l(x_t, u_t)$ represents the loss function of the linear dynamic system at time step t, and $\sum_{t=1}^{T} l(x_t, u_t)$ describes the cumulative loss over T time steps of the trajectory τ.

IV. METHOD

A. Cognitive Reasoning Trajectory Optimization

To mitigate the state space in policy search and accelerate policy learning, we introduce a novel LLMs-enhanced trajectory optimization method named CRTO. This approach integrates LLMs's logical reasoning abilities into the learning of reinforcement learning policies, thereby enhancing the quality of trajectory samples and enabling the agent to operate in a higher reward space.

As illustrated in Figure 1, our algorithm is delineated into two phases: the low-level cognitive control tuning stage and the trajectory optimization under unknown dynamics stage. In the first stage, we initially define the information exchange protocol between the robot and the LLM. Subsequently, a small set of incomplete trajectory samples from control task examples that conform to this information exchange protocol are fed to the LLM model for fine-tuning on specific tasks. For an MDP task such as 'close window', the fine-tuned LLM model directly outputs a single-step action value based on the robot's current state and feedback information. Given the stochastic nature of LLMs outputs, any non-standard responses are processed. In this stage, all states, actions, and relevant data are stored in a memory module. Repeating this single-step decision process yields a trajectory from the LLM that rapidly approaches the task objective. In the second stage, the suboptimal trajectories obtained are utilized to fit a dynamic model, which then informs the optimization of an LQR policy. Further, this LQR policy is sampled within the environment. The generated samples are used to refine the LQR policy. This process is iterated to learn the optimal Linear Gaussian policy. These processes will be elaborated in subsequent subsections.

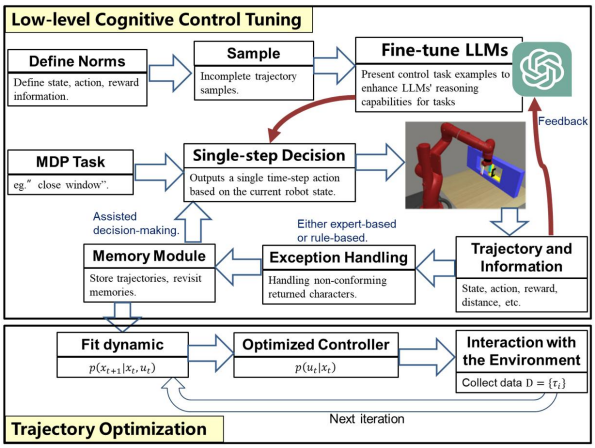

Fig. 1. The flowchart of the proposed Cognitive Reasoning Trajectory Optimization (CRTO) method.

B. Low-level Cognitive Control Tuning

LLMs are world models trained from vast amounts of internet data, analogous to pre-trained models in reinforcement learning tasks [31]. We define the complete world model of LLMs as f with parameter space Φ. In low-level robotic control problems, this model represents a mapping from the robot's state space $\mathcal{X}$ to its action space $\mathcal{U}$, denoted as $f_\Phi : \mathcal{X} \to \mathcal{U}$.

However, LLMs equipped solely with world model parameters cannot directly adapt to specific robotic operation tasks due to the absence of task-specific control information [32]. When LLMs encounter a new robotic learning task $\mathcal{T}_i$, it becomes necessary to fine-tune the world model parameters. Fine-tuning essentially involves adjusting the LLM's world model parameters Φ based on the task objectives, resulting in updated parameters Φ_i'. In MDP tasks, the objective of fine-tuning the LLM is to enable the robot to achieve optimal performance on the task using only a small number of task-related samples. This process is typically formalized as maximizing cumulative rewards or minimizing deviations from the task objectives. Defining the sample horizon length for fine-tuning as H, the loss of the LLM with parameters f_Φ on task $\mathcal{T}_i$ can be defined as the expected value over samples:

$$L_{\mathcal{T}_i}(f_\Phi) = \mathbb{E}_{x_t, u_t \sim f_\Phi, p(\tau)} \left[\sum_{t=1}^{H} L_{\mathcal{T}_i}(x_t^{LLMs}, u_t^{LLMs}) \right] \tag{2}$$

where x_t^{LLMs} and u_t^{LLMs} are states and actions generated according to the LLM parameters f_Φ and trajectory distribution $p(\tau)$. $L_{\mathcal{T}_i}(x_t, u_t)$ is the loss function of the robot's action samples under task $\mathcal{T}_i$, and H is the length of the sample sequence used for fine-tuning. Therefore, the objective of fine-tuning the LLM can be defined as learning the parameters that minimize the loss $L_{\mathcal{T}_i}(f_\Phi)$:

$$\Phi_i' = \arg\min_{\Phi} L_{\mathcal{T}_i}(f_\Phi), \quad i \in \mathbb{N} \tag{3}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The core technology of the LLM is the Transformer model, a type of deep neural network [21] [33]. Consequently, the parameter update process of the LLM can be described using a gradient descent model, and the parameters are updated via the gradient descent algorithm:

$$\Phi_i' \leftarrow \Phi - \alpha \nabla_{\Phi} L_{\mathscr{T}_i}(f_{\Phi}), \quad i \in \mathbb{N} \tag{4}$$

where $\nabla_{\Phi} L_{\mathscr{T}_i}(f_{\Phi})$ is the gradient of the loss function with respect to the world model parameters Φ, and α is the learning rate, fixed as a hyperparameter during the fine-tuning phase. $\mathbb{N}$ denotes the set of natural numbers.

In the following sections, we introduce the methodological details of the CRTO algorithm in three parts: defining task specifications, fine-tuning the LLM and updating the memory module, and exception handling.

a) Defining task specifications: To enable LLMs to comprehend its role and the tasks at hand, a textual description of the physical world's model is necessary. This textual modeling directly influences LLMs's understanding and reasoning capabilities regarding the operational tasks. For clear task articulation, we propose five categories of prompts for LLMs: Role Prompts (describing the role of LLMs), State Space Prompts (detailing the accessible state space and dimensions of the robot), Action Space Prompts (defining the dimensions and meanings of the robot's actions), Feedback Information Prompts (describing information fed back to LLMs from the environment), and Task Prompts (detailing the task requirements). The format for message exchange between LLMs and the environmental world during interaction is also presented. For instance, the message received by the robot from LLMs at time step t is:

$$\text{msg}_t^{\text{receive}} = \{u_t\} \tag{5}$$

where u_t represents the incremental movements of a Sawyer robot's end effector in Cartesian space (3 dimensions) and the opening/closing of the gripper (1 dimension). The feedback message from the robot to LLMs after executing actions in the operational environment is:

$$\text{msg}_t^{\text{send}} = \{x_t, R_t, d_t^{\text{reach}}, d_t^{\text{goal}}, \text{SR}_t, \text{info}_t\} \tag{6}$$

where x_t is the robot's state at step t, R_t is the reward obtained after executing action u_t, d_t^{reach} and d_t^{goal} are the distances from the robot's Tool Center Point (TCP) to the operational object and goal respectively, SR_t represents the success rate of the task at time t, and info_t encompasses other potentially useful information obtained from the environment.

In an MDP task, the trajectory rollout between the robot and LLMs takes the following form:

$$\tau_{\text{rollout}} = \{\text{msg}_0^{\text{send}}, \text{msg}_0^{\text{receive}}, \ldots, \text{msg}_t^{\text{send}}, \text{msg}_t^{\text{receive}}\} \tag{7}$$

In order to facilitate understanding, we provide herein a five-part example of prompts for invoking LLMs.

Role Prompts: *As an agent, you are currently controlling a 7-degree-of-freedom Sawyer robot.*

State Space Prompts: *You receive a 14-dimensional observation vector representing the state, consisting of joint angles (7 dimensions) and joint velocities (7 dimensions). For example: "state = [2.027, -0.567, -1.13, 1.731, 0.982, 1.12, 2.382, -4.036, 0.414, 4.067, -2.985, 0.866, -2.182, -1.677]".*

Action Space Prompts: *You can control the robot through a 4-dimensional action vector, which includes the positional changes of the robot's end-effector in Cartesian space (3 dimensions) and the incremental value of the gripper actuator (1 dimension). The output action values range from [-1.0, 1.0]. For example: "action = [0.22, 0.33, 0.56, 0.01]".*

Feedback Information Prompts: *Your output action values will be sent to the Sawyer robot. After the robot executes the action, the environment provides feedback information such as: "reward = -0.056, goalDist = 0.200, reachDist = 0.056, successRate = 0.0".*

Task Prompts: *Your objective is to minimize reach-dist to 0 as quickly as possible, and then minimize goal-dist to 0 as quickly as possible. Each time, we will send you the state information in the following format: "state = [2.027, -0.567, -1.13, 1.731, 0.982, 1.12, 2.382, -4.036, 0.414, 4.067, -2.985, 0.866, -2.182, -1.677], reward = -0.177, goalDist = 0.05, reachDist = 0.177, successRate = 0.0". You will return an action in the following format: "action = [0.22, 0.33, 0.56, 0.01]". For each query, you only need to return the action values without generating any additional characters.*

b) Fine-tuning the LLM and updating the memory module: In the user manual for GPT, it is noted that fine-tuning on a task with approximately 50-100 samples can enable large models to gain an understanding of the task. In our MDP task, a complete MDP trajectory contains about 10M data points. Thus, for 50-100 samples, the fine-tuning data volume would amount to approximately 500-1000M tokens. This substantial token consumption translates into high costs. Consequently, we did not adopt this direct fine-tuning approach in our task. Instead, we incorporated a Hybrid Memories module for indirect fine-tuning within the framework of LLMs. Our experiments demonstrated favorable outcomes with this setup.

In this study, we first populate the memory module with a modest amount of task-relevant prior demonstration samples D_{dem}. It is important to note that these expert trajectories span only about 20 to 30 timesteps and are not optimal. Primarily, these trajectories serve to demonstrate the standardization of input and output strings for LLMs. After each execution of actions derived from LLMs, the resulting state-action samples are added to the memory module's historical trajectories D_{mem}. For subsequent outputs from LLMs, all historical trajectories are fed back into the LLMs. We have observed that this configuration significantly enhances the LLMs' reasoning capabilities for specialized tasks. The trajectories in the memory module are also utilized for fitting dynamic models and trajectory optimization, which will be further discussed in subsequent sections.

c) Exception handling: Both syntactically and semantically, the outputs from LLMs may exhibit errors or deviate from predefined formats. This issue tends to occur more frequently with earlier versions of GPT, although it is less prevalent in the advanced GPT-4-0613 model. Despite the presence of instructive samples and historical trajectories, these inaccuracies remain inevitable. To address this issue, we have established discriminative rules for output evaluation. When the output aligns with the predefined string specifications, the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

algorithm processes it automatically. Conversely, when the output does not meet these specifications, expert intervention is required for correction. Our experiments indicate that minimal corrections enable the LLMs to subsequently produce outputs that adhere to the established specifications. Additionally, due to the potential for unsafe outputs from LLMs, we have currently imposed certain engineering constraints on the robot, such as limiting the range of motion values, speed, and workspace. Future research on developing safety-oriented control strategies for LLMs would be a promising direction.

C. Fit Dynamic Model

The Linear Quadratic Regulator (LQR) algorithm assumes that the dynamics are linear and known. Therefore, prior to learning the optimal policy, it is necessary to fit the parameters of the system dynamics model. At the initial stages of iteration, the samples required for fitting the dynamic model are sourced from the dataset $D_{LLMs} = \{x_t^{LLMs}, u_t^{LLMs}, x_{t+1}^{LLMs}\}$. These prior samples, originating from LLMs, demonstrate superior trajectory performance and are thus utilized for fitting the system dynamics model.

As outlined in Section III, the dynamic model at timestep t is defined as a time-varying linear multivariate Gaussian model:

$$p(x_{t+1}|x_t, u_t) = N(f_{xt}x_t + f_{ut}u_t + f_{ct}, F_t) \tag{8}$$

Due to the detailed derivation process presented in [34], we omit the parameter derivation here. Instead, we directly provide the values of the parameters for the dynamic model. $[f_{xt}, f_{ut}] = \Sigma_{xu,xu}^{-1}\Sigma_{xu,x'}$, $f_{ct} = \mu_{x'} - F_{mt}\mu_{xu}$ and $F_t = \Sigma_{x',x'} - F_{mt}\Sigma_{xu,xu}F_{mt}^T$.

D. Learning Linear Gaussian Controller

The learning objective of the linear Gaussian policy p_θ is to minimize the total expected cost over a fixed-length trajectory.

$$p_\theta^* \leftarrow argminE_{p_\theta}[\Sigma_{t=1}^T l(x_t, u_t)] \tag{9}$$

Here, θ represents the policy parameters, and T denotes the trajectory length. To facilitate problem handling, we describe this time-varying linear Gaussian policy as the probability distribution $p_\theta(u_t|x_t) = N(K_t x_t + k_t; C_t)$. Here, K_t and k_t are the mean parameters of the Gaussian distribution, and C_t is the covariance parameter of the probability distribution. In scenarios where dynamics are linear and costs are quadratic, the derivation process for the linear Gaussian controller (LGC) can be found in paper [35] [36].After iterative optimization using the LQR, the local policy for tasks is obtained as:

$$p_\theta(u_t|x_t) = N(K_t x_t + k_t; Q_{u,ut}^{-1}) \tag{10}$$

where $K_t = -Q_{u,ut}^{-1}Q_{u,xt}$ and $k_t = -Q_{u,ut}^{-1}Q_{ut}$.

E. Scalability and Generalization

Many robotic manipulation tasks can be decomposed into two distinct phases: approaching and manipulating. Currently, the reasoning capabilities of LLMs have demonstrated effectiveness primarily during the initial approaching phase. However, in the subsequent manipulation phase–where the robot must perform complex interactions with the environment–LLM-based control methods fail to yield effective results. The CRTO uses LLMs for fast inference in the first phase to generate expert demonstrations, reducing the search space in the robot's initial learning. In the second phase, reinforcement learning enables effective interaction for complex tasks. Consequently, the CRTO can be naturally extended to robotic manipulation tasks that can be simplified into these two phases. Due to network latency and inference delays when interacting with LLMs, the use of velocity control or torque control modes is precluded. Therefore, any robotic platform that supports position control can theoretically deploy the proposed CRTO.

In generalization, we distinguish between (a) generalizing to unexplored states within the state space and (b) generalizing to adjacent states. For (a), our method leverages high-quality state-action pairs from LLMs to narrow the initial policy search space, thereby improving learning efficiency. However, using good state-action pairs limits exploration and generalization to unknown states, indicating a trade-off between efficiency and broader generalization. Given this study's primary goal of enhancing efficiency, we do not further address out-of-distribution generalization. For (b), the proposed CRTO method represents the controller as a time-varying linear Gaussian probabilistic model, linearizing the robot's trajectory into T steps with local Gaussian models. This naturally enables the policy to generalize effectively to adjacent states, enhancing robustness to noise and overall reliability.

Algorithm 1: CRTO method

Input: initialization parameters
Output: $p_\theta(u_t|x_t)$

Define the state and action space, interaction protocols, and task details;
Fine-tune the LLMs model using a small number of demonstration samples D_{dem};
Store the demonstration samples into the memory module $D_{mem} \leftarrow D_{dem}$;
for *t = 1 to T* **do**
  Transmit the memory data and state information to the LLMs;
  The LLMs returns an action based on the state information;
  Process any non-compliant information;
  The robot executes the current timestep action and obtains new state, reward, distance, and other information;
  Store all information into the memory module D_{mem} ;
end
for *iter = 1 to N* **do**
  Fit dynamic model using D_{mem};
  Update the controller through trajectory optimization;
  Interact with the environment to update the sample pool D_{mem}.
end
Return optimized policy parameters θ^*.

F. Pseudocode

Algorithm 1 outlines the proposed CRTO algorithm, which comprises two main phases. Initially, the Low-level Cognitive Control Tuning technique is employed to swiftly extract and

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

comprehend deep features and patterns within the robot's state-action sequences, obtaining trajectory samples with task-specific priors that aid subsequent decision-making. Subsequently, these samples are used to fit a dynamic model and conduct trajectory optimization. Iterative updates to the controller are performed until a specific control strategy for the designated robotic manipulation task is learned.

V. EXPERIMENTS

A. Experimental Setup

Experiments were conducted on three Sawyer robot manipulation tasks within the Continual World [37] [38], utilizing the Mujoco simulator. These tasks included: (a) button-press-v1, where the objective is to press a button to a specified position; (b) drawer-close-v1, which involves the robot closing a drawer; and (c) close-window-v1, where the robot pushes a handle to fully close a window. The deployment of our algorithm involves two primary computational resources: local computing resources and cloud-based inference resources. Local computations were performed using an Intel Core i7-8700 CPU operating at 3.2 GHz. For 10 iterations, the training time was approximately 2 hours. The cloud computations utilized the inference capabilities of ChatGPT, accessed via its API interface. As OpenAI has not disclosed the computational requirements or technical details of ChatGPT, the actual computational resource consumption during inference is unknown. When training policy using the CRTO method, our proposed LCCT technique consumes approximately 6,000 to 8,000 tokens per interaction with the ChatGPT API. Depending on network latency and inference congestion, each interaction with the LLM takes approximately 2 to 5 seconds.

In the experiment, the algorithm interacts with the Mujoco via socket communication at a frequency of 10Hz. The length of the sampled trajectory is 200 time steps. The state space comprised joint angles (7 dimensions) and joint velocities (7 dimensions), while the action space included 3D end-effector position changes and gripper actuator delta. For other simulation environments (e.g., Gazebo) or deployment on real robots, communication between the high-level and low-level systems can be facilitated using the $<rosservice>$ and $<rostopic>$ mechanisms in ROS. The high-level system runs the algorithms and invokes LLMs through the APIs provided by OpenAI, while the low-level system is responsible for executing actions within the environment and providing feedback on environmental information. The objective function for these manipulation tasks was defined as follows.

$$l(x_t, u_t) = w_u l(u_t) + w_{reach} l(d_{r,t}) + w_{goal} l(d_{g,t}) \quad (11)$$

It consists of three components: the action cost $l(u_t)$, the distance cost between the TCP and the manipulated object $l(d_{r,t})$, and the distance cost between the manipulated object and the target position $l(d_{g,t})$. The weights for these costs, w_u, w_{reach}, and w_{goal}, were all set to 1.0. The distance cost function is calculated as $l(d_t) = l_1 d_t^2 + l_2 \log(d_t^2 + \alpha)$, where α is a small constant set to 0.0001 to prevent invalid logarithmic values. l_1 and l_2 balance the influence of large and small distances on the target, aiming to prevent the agent from converging to suboptimal local minima near the goal.

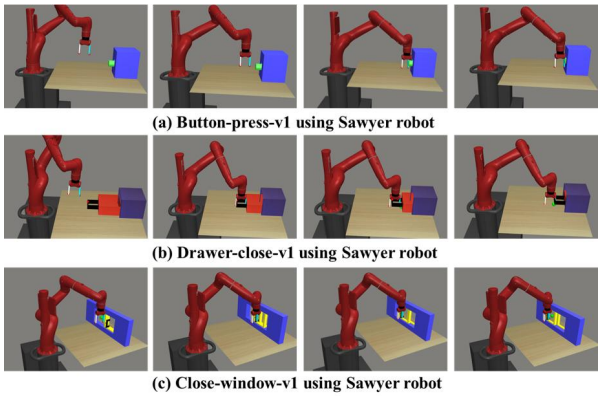

Fig. 2. Examples of experimental tasks. The left image depicts the initial state of the task, the middle image shows the intermediate state, and the right image illustrates the final state. The tasks were conducted using the MuJoCo simulator, and the scenarios are derived from the continue world environment.

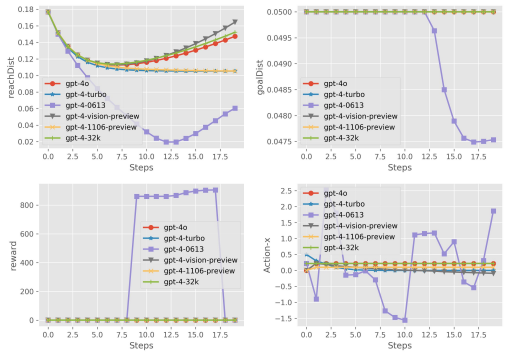

Fig. 3. In the iterative process, the performance of different GPT models, after fine-tuning, on robotic low-level control is examined. The top-left graph represents the distance between the robot's TCP and the object being manipulated; the top-right graph shows the distance of the manipulated object from the target; the bottom-left graph depicts the reward curve; and the bottom-right graph illustrates the action values in the Cartesian space along the x-direction output by the LLM at the current timestep.

B. Performance of LCCT Technology Across Various LLMs

Currently, OpenAI's ChatGPT is among the most representative LLMs. For our algorithm, employing ChatGPT or other open-source LLMs to collect expert demonstrations does not fundamentally differ. Provided an LLM's reasoning capabilities suffice to generate expert demonstrations during the approaching phase of robotic manipulation tasks, it can effectively accelerate trajectory optimization training. Therefore, we evaluated the performance of LCCT using various GPT models. After multiple rounds of testing, we identified that the GPT-4-0613 has the strongest reasoning capabilities. Our assessments were conducted on the button-press-v1 task.

Figure 3 illustrates the results. The four sets of results include $reachDist$, $goalDist$, $reward$, and the $action$ in the X-direction within Cartesian space. As shown in the top left graph, the LCCT method incorporating the GPT-4-0613 model rapidly approaches the target location. In just 13 time steps, the robot's TCP approached the manipulated object, a result unachievable by all other reinforcement learning methods. Other models, although initially capable of gradually moving the robot's end-effector closer to the object, diverged after approximately 8 rounds. The upper right figure shows that only the strategy employing the GPT-4-0613 model successfully facilitated movement of the object, though it reduced the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

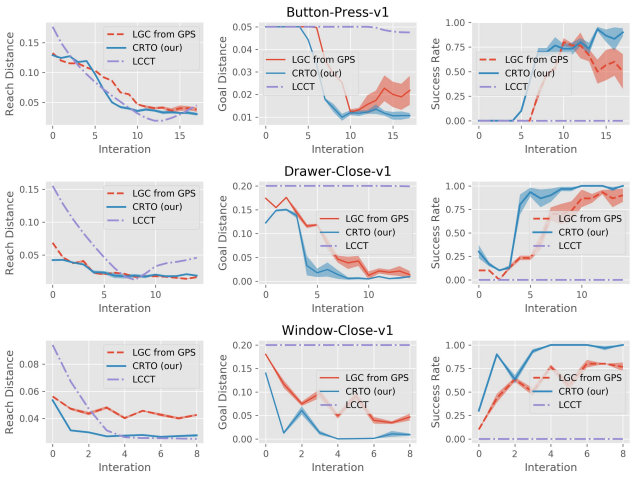

Fig. 4. The performance of three methods across various robotic manipulation tasks is presented. Each task demonstrates the iterative progression of reach distance, goal distance, and success rate.

$goalDist$ by merely approximately 0.0025 meters. These results validate the feasibility of directly applying LCCT techniques with LLMs for low-level MDP control tasks in robotics. Based on these findings, we employed the GPT-4-0613 model for logical reasoning in all subsequent experiments.

C. Comparison with Other Methods

To validate the policy search efficiency of CRTO, we conducted comparative experiments against LGC, which is derived from the GPS algorithm?a representative model-based method. In this section, we tested on three manipulation tasks within the Continual World. The variations in $reachDist$, $goalDist$, and $successRate$ during the iterative process of the algorithms are depicted in Figure 4. The performance metrics for both CRTO and LGC from the GPS algorithm in Figure 4 represent the results of three repeated experiments. The single experimental result using only LCCT to control LLMs for completing manipulation tasks is due to our findings during adjustments to the ”temperature” parameter of the control LLMs model, which modulates the randomness of generated text. We observed that at values less than a certain threshold (e.g., $0 \leq$ temperature ≤ 0.355), the actions generated at each timestep were nearly identical, whereas above a critical value (e.g., 0.355), the actions became ineffective in the operational environment. Consequently, we did not include error bars for the LLM model-generated distance curves in the figure. It is important to note that each iteration on the graph for LCCT corresponds to each timestep.

The experimental results, as shown in Figure 4, demonstrate that the CRTO method achieves a faster convergence of the $goal - dist$ values than the LGC from GPS method across the three tasks. For instance, in the drawer-close-v1 task, the CRTO method's $goalDist$ rapidly converges to below 0.03 by the fourth iteration, whereas the LGC from GPS method approaches convergence around the tenth iteration. Moreover, the CRTO method approaches a success rate close to 1.0 more swiftly and achieves higher success rate values. The graph also shows that while LCCT technology allows the GPT model to quickly approach target positions, its execution is hindered by the LLM model's limited logical reasoning capabilities. The results from Figure 4 affirm the efficacy of the CRTO method, aligning with our design expectations.

D. Impact of LLMs Sample Proportions on Convergence Speed

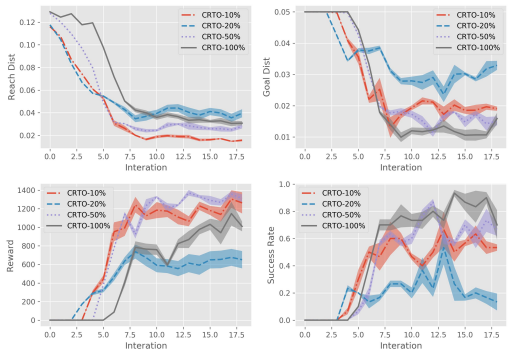

Fig. 5. Algorithm convergence curves under varying proportions of LLM prior samples. The results depict the numerical variation curves for reach distance, goal distance, reward, and success rate.

Due to the expert samples from LLMs comprising only a single incomplete trajectory of 20 time steps, the full trajectory length used during training is 200 time steps. To facilitate fitting the dynamic model and standardize trajectory lengths, this study employs a replay technique on the initial trajectory using these expert samples, with the remaining length supplemented by random samples. We investigate the impact of the replay ratio of LLM expert samples on training outcomes. Experiments were conducted on the button-press-v1 task, with each condition replicated thrice. We collected samples over 20 time steps using LLMs, which were then utilized to fit the dynamic model. As shown in Figure 5, the CRTO method exhibited the lowest goal distance and the highest success rate when the prior mixing ratio was 100%. We posit that an ample supply of prior samples facilitates a more accurate fitting of the dynamic model. The optimal proportion of prior samples requires careful adjustment during experimentation.

E. Comparison with Motion Planning Diffusion

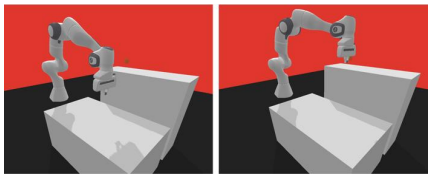

Fig. 6. Panda Shelf, as presented in the paper [39]

Motion Planning Diffusion (MPD) is a novel trajectory generation method that learns a diffusion-based trajectory generation model by leveraging expert trajectories. The method formulates motion planning as planning-as-inference, where sampling is performed from the posterior distribution using guidance from the diffusion model. As Fig. 6 shows, we replicated the Panda Shelf scenario in the Panda-gym environment as presented in the [39]. In Table 1, we directly quote the experimental results of the Gaussian Process Motion Planning (GPMP) and MPD methods in the Panda Shelf scenario, as reported in the paper [39]. Using the same experimental setup, we evaluated the CRTO algorithm in the same scenario with

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

ten different initial-to-goal configurations, each tested 100 times. Results are shown in Table 1. The success rate of the CRTO is approximately $94.9 \pm 11.29\%$. While this success rate is not the highest in the scenario, it is important to note that the paper [39] mentions the high cost of data collection, requiring the collection of 10,000 expert samples for a single task. In contrast, our methods only requires a single incomplete trajectory of approximately 20-30 time steps as prior input. Furthermore, obtaining expert demonstrations via the MPD required approximately 6 hours on a workstation equipped with an AMD EPYC 7453 28-Core processor and an NVIDIA GeForce RTX 3090 GPU, whereas our method required merely 40-100 seconds of inference time on OpenAI's cloud platform. Therefore, the CRTO achieves comparable performance with fewer prior samples and significantly reduced local data collection time, validating its effectiveness.

TABLE I
THE SUCCESS RATE IN THE PANDA SHELF SCENARIO.

-	GPMP	MPD	CRTO
Success Rate	$88.0 \pm 32.5\%$	$100 \pm 0.0\%$	$94.9 \pm 11.29\%$

VI. CONCLUSIONS

In this paper, we propose a reinforcement learning method for efficient robotic skill acquisition, termed CRTO. However, the algorithm does exhibit some dependence on the logical reasoning capabilities of LLMs. The enhancement of CRTO algorithmic learning efficiency relies on obtaining expert samples from LLMs. Only when LLMs possess sufficient logical reasoning capabilities – enabling them to adjust each action value based on feedback rewards and distance information – can the generated samples be of high quality. These high-quality samples are crucial for guiding the dynamic model fitting and trajectory optimization processes. Conversely, if LLMs lack reasoning abilities and produce only low-quality samples, it becomes impossible to steer the policy search into regions of high return. As these capabilities improve, CRTO is expected to learn even faster. Fine-tuning LLMs using abundant expert demonstrations from robotic manipulation tasks may potentially enhance their control performance in the second stage. Integrating LLMs with reinforcement learning marks a step toward general artificial intelligence.

REFERENCES

[1] K. Chatzilygeroudis, V. Vassiliades, F. Stulp *et al.*, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Trans. Robot.*, vol. 36, no. 2, pp. 328–347, 2019.

[2] S. Levine *et al.*, "Learning contact-rich manipulation skills with guided policy search," *in Proc. IEEE Int. Conf. Robot. Autom.*, 2015.

[3] A. Nagabandi *et al.*, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2018, pp. 7559–7566.

[4] W. Huang, C. Wang, R. Zhang *et al.*, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[5] A. Brohan *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *in Proc. CoRL*, pp. 287–318, 2023.

[6] Y. Shen *et al.*, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face," *in Proc. Neural Inf. Process. Syst.*, vol. 36, 2024.

[7] J. o. Wei, "Chain-of-thought prompting elicits reasoning in large language models," *NeurIPS*, vol. 35, pp. 24 824–24 837, 2022.

[8] J. Liang *et al.*, "Code as policies: Language model programs for embodied control," in *ICRA*. IEEE, 2023, pp. 9493–9500.

[9] W. Huang *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[10] Z. Xi, W. Chen, X. Guo *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.

[11] M. Cutler *et al.*, "Efficient reinforcement learning for robots using informative simulated priors," in *ICRA*. IEEE, 2015, pp. 2605–2612.

[12] Y. Chebotar, A. Handa, V. Makoviychuk *et al.*, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *in Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2019, pp. 8973–8979.

[13] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Proc. CoRL*. PMLR, 2017, pp. 334–343.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[15] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1360–1370, 2012.

[16] R. Pautrat, K. Chatzilygeroudis, and J.-B. Mouret, "Bayesian optimization with automatic prior selection for data-efficient direct policy search," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2018, pp. 7571–7578.

[17] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers," in *Proc. CoRL*. PMLR, 2017, pp. 47–56.

[18] A. Cully, J. Clune, D. Tarapore *et al.*, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[19] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *J. Intell. & Robotic Syst.*, vol. 86, no. 2, pp. 153–173, 2017.

[20] M. P. Deisenroth *et al.*, "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, no. 1–2, pp. 1–142, 2013.

[21] Y. Cao, H. Zhao, Y. Cheng *et al.*, "Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods," *arXiv preprint arXiv:2404.00282*, 2024.

[22] S. Yao *et al.*, "Tree of thoughts: Deliberate problem solving with large language models," *in Proc. Neural Inf. Process. Syst.*, vol. 36, 2024.

[23] M. Besta *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," *AAAI*, vol. 38, no. 16, pp. 17 682–17 690, 2024.

[24] M. Dalal *et al.*, "Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks," *arXiv preprint arXiv:2405.01534*, 2024.

[25] T. Silver, S. Dan, K. Srinivas *et al.*, "Generalized planning in pddl domains with pretrained large language models," in *AAAI*, vol. 38, no. 18, 2024, pp. 20 256–20 264.

[26] S. Wang, M. Han, Z. Jiao *et al.*, "Llm3: Large language model-based task and motion planning with motion failure reasoning," *arXiv preprint arXiv:2403.11552*, 2024.

[27] J. Wu *et al.*, "Tidybot: Personalized robot assistance with large language models," *Auton. Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.

[28] C. Tang, D. Huang, W. Ge *et al.*, "Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping," *IEEE Robot. Autom. Lett.*, 2023.

[29] T. Kwon, N. Di Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robot. Autom. Lett.*, 2024.

[30] L. Chen, Y. Lei, S. Jin *et al.*, "Rlingua: Improving reinforcement learning sample efficiency in robotic manipulations with large language models," *IEEE Robot. Autom. Lett.*, 2024.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," pp. 1126–1135, 2017.

[32] W. Huang, P. Abbeel, D. Pathak *et al.*, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *ICML*. PMLR, 2022, pp. 9118–9147.

[33] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[34] Q. Dong *et al.*, "Kalman filter-based one-shot sim-to-real transfer learning," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 311–318, 2023.

[35] W. Han *et al.*, "Learning compound multi-step controllers under unknown dynamics," in *IROS*. IEEE, 2015, pp. 6435–6442.

[36] S. Levine, C. Finn *et al.*, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.

[37] M. Woczyk, *et al.*, "Continual world: A robotic benchmark for continual reinforcement learning," *NeurIPS*, vol. 34, pp. 28 496–28 510, 2021.

[38] T. Yu, D. Quillen, Z. He *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *in Proc. CoRL*. PMLR, 2020, pp. 1094–1100.

[39] J. Carvalho, A. T. Le, M. Baierl *et al.*, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *in Proc. Int. Conf. Intell. Robots Syst.* IEEE, 2023, pp. 1916–1923.