Learning Compositional Behaviors from Demonstration and Language

Weiyu Liu[1*], **Neil Nie**[1*], **Ruohan Zhang**[1], **Jiayuan Mao**[2†], **Jiajun Wu**[1†]
[1]Stanford University [2]MIT

Abstract: We introduce Behavior from Language and Demonstration (BLADE), a framework for long-horizon robotic manipulation by integrating imitation learning and model-based planning. BLADE leverages language-annotated demonstrations, extracts abstract action knowledge from large language models (LLMs), and constructs a library of structured, high-level action representations. These representations include preconditions and effects grounded in visual perception for each high-level action, along with corresponding controllers implemented as neural network-based policies. BLADE can recover such structured representations automatically, without manually labeled states or symbolic definitions. BLADE shows significant capabilities in generalizing to novel situations, including novel initial states, external state perturbations, and novel goals. We validate the effectiveness of our approach both in simulation and on a real robot with a diverse set of objects with articulated parts, partial observability, and geometric constraints.

Keywords: Manipulation, Planning Abstractions, Learning from Language

1 Introduction

Developing autonomous robots capable of completing long-horizon manipulation tasks is a significant milestone. We want to build robots that can directly perceive the world, operate over extended periods, generalize to various states and goals, and are robust to perturbations. A promising direction is to combine learned policies with model-based planners, allowing them to operate on different time scales. In particular, imitation learning-based methods have proven highly successful in learning policies for various "behaviors," which usually operate over a short time span [e.g., 1]. To solve more complex and longer-horizon tasks, we can compose these behaviors by planning in abstract action spaces [2–4], in latent spaces [5], or via large pre-trained models such as large language models [6].

However, one of the key challenges of all high-level planning approaches is the automatic acquisition of an abstraction for the learned "behaviors" to support long-horizon planning. The goal of this behavior abstraction learning is to build representations that describe the preconditions and effects of behaviors, to enable chaining and search. These representations should depend on the environment, the set of possible goals, and the specifications of individual behaviors. Furthermore, these representations should be grounded on high-dimensional perception inputs and low-level robot control commands.

Our insight into tackling this challenge is to leverage knowledge from two sources: the low-level, mechanical understanding of robot-object contact, and the high-level, abstract understanding of object-object interactions described in language that can be extracted from language models as the knowledge source. Our framework, behavior from language and demonstration (BLADE), takes as input a small number of language-annotated demonstrations (Fig. 1a). It segments each trajectory based on which object is in contact with the robot. Then, it uses a large language model (LLM), conditioned on the contact sequences and the language annotations, to propose abstract behavior descriptions with preconditions and effects that best explain the demonstration trajectories. During training, we extract the state abstraction terms from the preconditions and effects (e.g., *turned-on*,

[*] denotes equal contribution. [†] denotes equal advising. Project page and videos: https://blade-bot.github.io/.

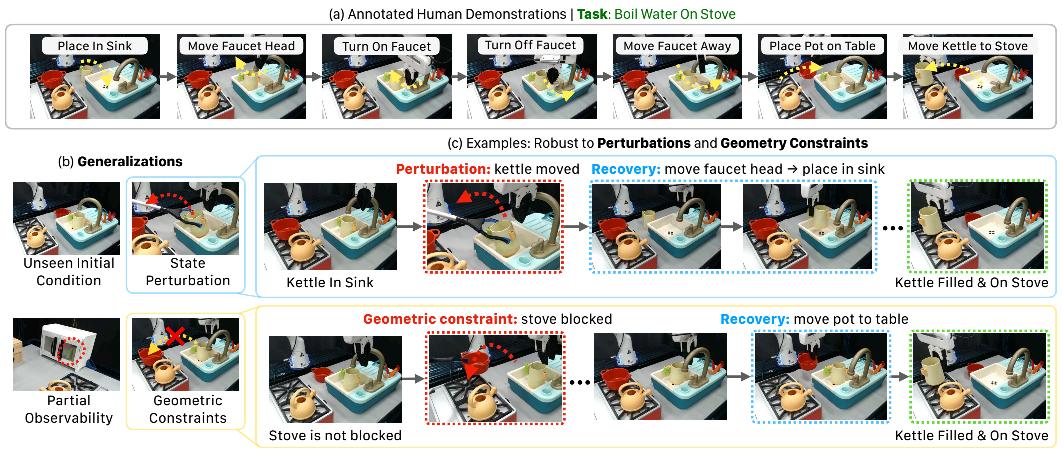


Figure 1: BLADE, a robot manipulation framework combining imitation learning and model-based planning. (a) BLADE takes language-annotated demonstrations as training data. (b) It generalizes to unseen initial conditions, state perturbations, and geometric constraints. (c) In the depicted scenarios, BLADE recovers from perturbations such as moving the kettle out of the sink, and resolves geometric constraints including a blocked stove.

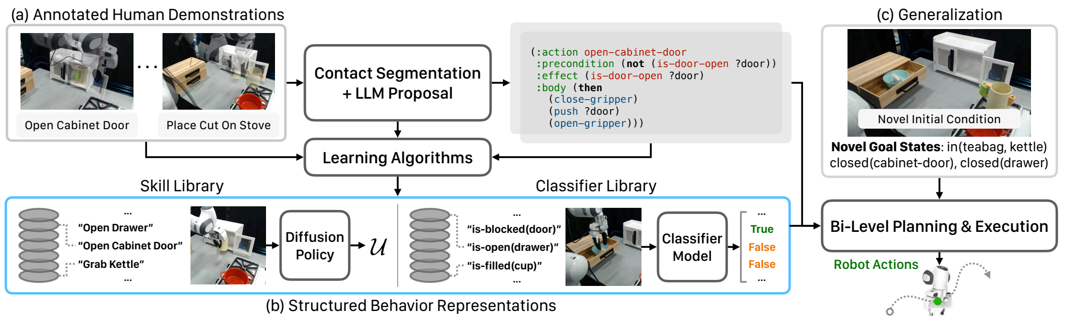
aligned-with), and learn their groundings on perception inputs. We also learn the control policies associated with each behavior (e.g., *turn on the faucet*).

Our model offers several advantages. First, unlike prior work that relies on manually defined state abstractions or additional state labels, our method automatically generates state abstraction labels based on the language-annotated demonstrations and LLM-proposed behavior descriptions. BLADE recovers the visual grounding of these abstractions without any additional label. Second, BLADE generalizes to novel states and goals by composing learned behaviors using a planner. Shown in Fig. 1b, it can handle various novel initial conditions and external perturbations that lead to unseen states. Third, our method can handle novel geometric constraints (Fig. 1c) and partial observability from articulated bodies like drawers.

2 Related Work

Composing skills for long-horizon manipulation. A large body of model-based planning methods use manually-defined transition models [2, 7–12] or models learned from data [13–18] to generate long-horizon plans. However, learning dynamics models with accurate long-term predictions and strong generalization remains challenging. A related direction is to introduce hierarchical structures into the policy models [19–25], where different methods can segment continuous demonstrations into short-horizon skills [23, 26, 27]. Facing the challenges in modeling action dependencies, these methods are limited to following sequentially specified subgoals. Some work addresses this issue by learning the dependencies between actions from data, but they require large-scale supervised datasets [28–31]. Our approach is related to methods that learn symbolic action representations [32–36]; the difference is that BLADE uses a LLM to generates causal models of the environment and learns their groundings on sensory inputs.

Using LLMs for planning. Many researchers have explored using LLMs for planning. Methods for direct generation of action sequences [37, 38] can struggle to produce accurate plans [39, 40]. Researchers have also leveraged LLMs as translators from natural language instructions to symbolic goals [41–44], as generalized solvers [45], as memory modules [46], and as world models [47, 48]. To improve the planning accuracy, prior work has explored techniques including using programs [49, 50], learning affordance functions [6, 51], replanning [52], finetuning [53–55], embedding reasoning in a behavior tree [56], and VLM-based decision-making [57, 58]. BLADE shares a similar spirit as methods using LLMs to generate planning-compatible action representations [59–61]. However, they make assumptions on the availability of state abstractions, while BLADE grounds LLM-generated action definitions without additional labels. Also complementary to methods that leverage these representations for skill learning [62, 63], our approach uses them for composing skills in novel ways.



```
(:action open-cabinet-door
 :precondition (not (is-door-open ?door))
 :effect (is-door-open ?door)
 :body (then
   (close-gripper)
   (push ?door)
   (open-gripper)))
```

Figure 2: Overview of BLADE. (a) BLADE receives language-annotated human demonstrations, (b) segments demonstrations into contact primitives, and learns a structured behavior representation. (c) It generalizes to novel conditions by leveraging bi-level planning and execution to achieve goal states.

3 Problem Formulation

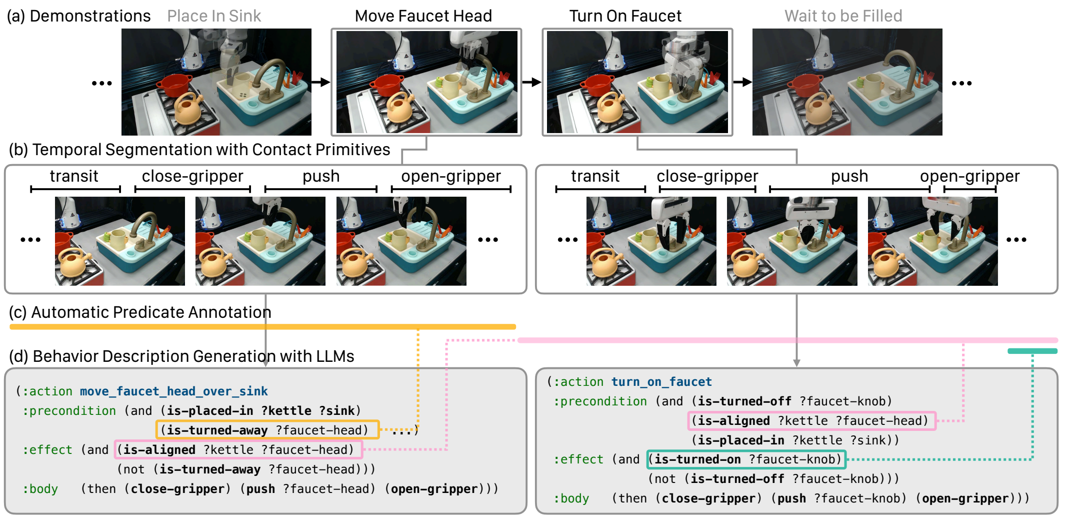
We consider the problem of learning a language-conditioned goal-reaching manipulation policy. Formally, the environment is modeled as a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{T} \rangle$ where $\mathcal{X}$ is the raw state space, $\mathcal{U}$ is the low-level action space, and $\mathcal{T} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the transition function (which may be stochastic and unknown). Furthermore, the robot will receive observations $o \in \mathcal{O}$ that may be partially observable views of the states. At test time, the robot also receives a natural language instruction ℓ_t, which corresponds to a set of goal states. An oracle goal satisfaction function defines whether the language goal is reached, i.e., $g_{\ell_t} : \mathcal{X} \to \{T, F\}$. Given an initial state $x_0 \in \mathcal{X}$ and the instruction ℓ_t, the robot should generate a sequence of low-level actions $\{u_1, u_2, ..., u_H\} \in \mathcal{U}^H$.

In the language-annotated learning setting, the robot has a dataset of language-annotated demonstrations $\mathcal{D}$. Each demonstration is a sequence of robot actions $\{u_1, ..., u_H\}$ paired with observations $\{o_0, ..., o_H\}$. Each trajectory is segmented into M sub-trajectories, and natural language descriptions $\{\ell_1, ..., \ell_M\}$ are associated with the segments (e.g., "*place the kettle on the stove*"). In this paper, we assume that there is a finite number of possible ℓ's—each corresponding to a skill to learn.

Directly learning a single goal-conditioned policy that can generalize to novel states and goals is challenging. Therefore, we recover an *abstract* state and action representation of the environment and combine online planning in abstract states and offline policy learning for low-level control to solve the task. In BLADE, behaviors are represented as temporally extended actions with preconditions and effects characterized by state predicates. Formally, we want to recover a set of predicates $\mathcal{P}$ that define an abstract state space $\mathcal{S}$. We focus on a scenario where all predicates are binary. However, they are grounded on high-dimensional sensory inputs. Using $\mathcal{P}$, a state can be described as a set of grounded atoms such as $\{\textit{kettle}(\text{A}), \textit{stove}(\text{B}), \textit{filled}(\text{A}), \textit{on}(\text{A}, \text{B})\}$ for a two-object scene. BLADE will learn a function $\Phi : \mathcal{O} \to \mathcal{S}$ that maps observations to abstract states. In its current implementation, BLADE requires humans to additionally provide a list of predicate names in natural language, which we have found to be helpful for LLMs to generate action definitions. We provide additional ablations in the Appendix A.2. Based on $\mathcal{S}$, we learn a library of *behaviors* (a.k.a., *abstract actions*). Each behavior $a \in \mathcal{A}$ is a tuple of $\langle \textit{name}, \textit{args}, \textit{pre}, \textit{eff}, \pi \rangle$. *name* is the name of the action. *args* is a list of variables related to the action, often denoted by $?x, ?y$. *pre* and *eff* are the precondition and effect formula defined in terms of the variables *args* and the predicates $\mathcal{P}$. A low-level policy $\pi : \mathcal{O} \to \mathcal{U}$ is also associated with a. The semantics of the preconditions and effects is: for any state x such that $\textit{pre}(\Phi(x))$ is satisfied, executing π at x will lead to a state x' such that $\textit{eff}(\Phi(x'))$ [64].

4 Behavior from Language and Demonstration

BLADE is a method for learning abstract state and action representations from language-annotated demonstrations. It works in three steps, as illustrated in Fig. 2. First, we generate a symbolic behavior definition conditioned on the language annotations and contact sequences in the demonstration using a large language model (LLM). Next, we learn the classifiers associated with all state predicates and the control policies, all from the demonstration without additional annotations. At test time, we use a bi-level planning and execution strategy to generate robot actions.



```
(:action move_faucet_head_over_sink
 :precondition (and (is-placed-in ?kettle ?sink)
                    (is-turned-away ?faucet-head)  ...)
 :effect (and (is-aligned ?kettle ?faucet-head)
              (not (is-turned-away ?faucet-head)))
 :body   (then (close-gripper) (push ?faucet-head) (open-gripper)))
```

```
(:action turn_on_faucet
 :precondition (and (is-turned-off ?faucet-knob)
                    (is-aligned ?kettle ?faucet-head)
                    (is-placed-in ?kettle ?sink))
 :effect (and (is-turned-on ?faucet-knob)
              (not (is-turned-off ?faucet-knob)))
 :body   (then (close-gripper) (push ?faucet-knob) (open-gripper)))
```

Figure 3: Behavior Descriptions Learning. (a) A demonstration is provided along with corresponding language annotations. (b) The demonstration is segmented into a sequence of contact primitives. (c) A large language model interprets the annotation and contact sequence, generating a symbolic behavior definition. (d) The system automatically generates data to learn classifiers for state predicates.

4.1 Behavior Description Learning

Given a finite set of behaviors with language descriptions $\{\ell\}$ and corresponding demonstration segments, we generate an abstract description for each ℓ by querying large language models. To facilitate LLM generation, we provide additional information on the list of objects with which the robot has contact. The generated operators are further refined with abstract verification.

Temporal segmentation. We first segment each demonstration (Fig. 3a) into a sequence of *contact-based primitives* (Fig. 3b). In this paper we consider seven primitives describing the interactions between the robot and other objects: *open*/*close* grippers without holding objects, *move-to*(x) which moves the gripper to an object, *grasp*(x, y) and *place*(x, y) which grasp and place object x from/onto another object y, *move*(x) which moves the currently holding object x and *push*(x). We leverage proprioception, i.e., gripper open state, and object segmentation to automatically segment the continuous trajectories into these basis segments. For example, pushing the faucet head away involves the sequence of $\{$*close-gripper*, *push*, *open-gripper*$\}$. This segmentation will be used for LLMs to generate operator definitions and for constructing training data for control policies.

Behavior description generation with LLMs. Our behavior description language is based on PDDL [65]. We extend the PDDL definition to include a *body* section which is a sequence of contact primitives. It will be generated by the LLM based on the demonstration data.

Our input to the LLM mainly contains: 1) a general description of the environment, 2) the natural language descriptions ℓ associated with the behavior itself and other behaviors that have appeared preceding or following ℓ in the dataset, 3) all possible sequence of contact primitive sequences associated with ℓ across the dataset, and 4) additional instructions on the PDDL syntax, including a single PDDL definition example. We find the additional context useful. As shown in Fig. 3d, in addition to preconditions and effects of the operators, we also ask LLMs to predict a *body* of contact primitive sequence associated with the behavior, which we call *body*. We assume that each behavior has a single corresponding contact primitive sequence, and use this step to account for noises in the segmentation annotations. After LLM predicts the definition for all behavior, we will re-segment the demonstrations associated with each behavior based on the LLM-predicted body section.

Behavior description refinement with abstract verification. In addition to checking for syntax errors, we also verify the generated behavior descriptions with *abstract verification* on the demonstration trajectories. Given a segmented sequence of the trajectory where each segment is associated with a behavior, we verify whether the preconditions of each behavior can be satisfied by the accumulated effects of the previous segments. This verification does not require learning the grounding of predicates and can be done at the behavior level for incorrect preconditions and effects, and at the contact primitive level for missing or incorrect contact primitives (e.g., *grasp* cannot be immediately followed by other *grasp*). We resample behavior definitions that do not pass the verification.

4.2 Classifier and Policy Learning

Given the dataset of state-action segments associated with each behavior, we train the classifiers for different state predicates and the low-level controller for each behavior.

Automatic predicate annotation. We leverage *all* behavior descriptions to automatically label an observation sequence $\{o_1, ..., o_H\}$ based on its associated segmentation. In particular, at o_0, we label all state predicates as "unknown." Next, we unroll the sequence of behavior executed in $\bar{o}$. As illustrated in Fig. 3c, before applying a behavior a at step o_t, we label all predicates in pre_a true and predicates in eff_a false. When a finishes at step $o_{t'}$, we label all predicates in eff_a. In addition, we will propagate the labels for state predicates to later time steps until they are explicitly altered by another behavior a. In contrast to earlier methods, such as Migimatsu and Bohg [66] and Mao et al. [67], which directly use the first and last state of state-action segments to train predicate classifiers, our method greatly increases the diversity of training data. After this step, for each predicate $p \in \mathcal{P}$, we obtain a dataset of paired observations o and the predicate value of p at the corresponding time step.

Classifier learning. Based on the state predicate dataset generated from behavior definitions, we train a set of state classifiers $f_\theta(p) : \mathcal{O} \to \{T, F\}$, which are implemented as standard neural networks for classification. We include implementation details in Appendix A.6. In real-world environments with strong data-efficiency requirements, we additionally use an open vocabulary object detector [68] to detect relevant objects for the state predicate and crop the observation images. For example, only pixels associated with the object faucet will be the input to the *turned-on*(faucet) classifier.

Policy learning. For each behavior, we also train control policies $\pi_\theta(a) : \mathcal{O} \to \mathcal{U}$, implemented as a diffusion policy [1]. In simulation, we use a combination of frame-mounted and wrist-mounted RGB-D cameras as the inputs to the diffusion policy, while in the real world, the policy takes raw camera images as input. The high-level planner orchestrates which of these low-level policies to deploy based on the scene and states. Once trained on these diverse demonstrations of different skills, the resulting low-level policies can adapt to local changes, such as variations in object poses.

4.3 Bi-Level Planning and Execution

At test time, given a novel state and a novel goal, BLADE first uses LLMs to translate the goal into a first-order logic formula based on the state predicates. Next, it leverages the learned state abstractions to perform planning in a symbolic space to produce a sequence of behaviors. Then, we execute the low-level policy associated with the first behavior, and we re-run the planner after the low-level policy finishes—this enables us to handle various types of uncertainties and perturbations, including execution failure, partial observability, and human perturbation. In implementation, we use the fast-forward heuristic to generate plans [69]; however, our method is planner-agnostic, and other symbolic planners (e.g., Fast-Downward [70]) are compatible.

Visibility and geometric constraints are also modeled as preconditions, in addition to other object-state and relational conditions. For example, the behavior "opening the cabinet door" will have preconditions on the initial state of the door, a visibility constraint that the door is visible, and a geometric constraint that nothing is blocking the door. When those preconditions are not satisfied, the planner will automatically generate plans, such as actions that move obstacles away, to achieve them. Partial observability was handled by using the most-likely state assumption during planning and performing replanning. We include details in Appendix A.8.

5 Experiments

5.1 Simulation Experimental Setup

We use the CALVIN benchmark [71] for simulation-based evaluations, which include teleoperated human-play data. We use the split D of the dataset, which consists of approximately 6 hours of interactions. Annotations of the play data are generated by a script that detects goal conditions

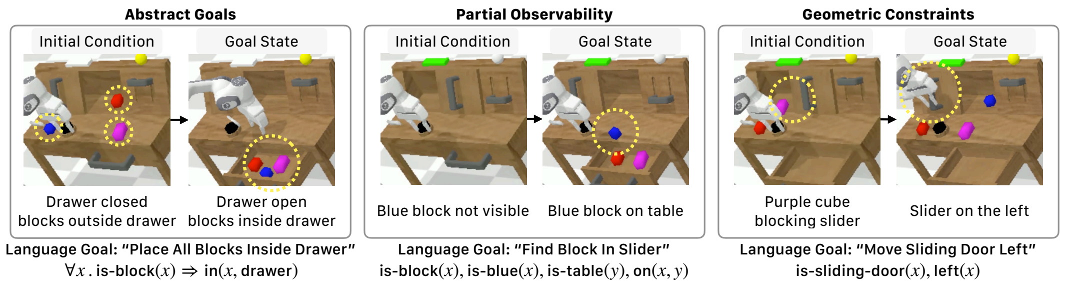

Abstract Goals
Initial Condition
Goal State
Drawer closed blocks outside drawer
Drawer open blocks inside drawer
Language Goal: "Place All Blocks Inside Drawer"
∀x . is-block(x) ⇒ in(x, drawer)
Partial Observability
Initial Condition
Goal State
Blue block not visible
Blue block on table
Language Goal: "Find Block In Slider"
is-block(x), is-blue(x), is-table(y), on(x, y)
Geometric Constraints
Initial Condition
Goal State
Purple cube blocking slider
Slider on the left
Language Goal: "Move Sliding Door Left"
is-sliding-door(x), left(x)

Figure 4: Generalization Tasks in CALVIN. Examples from the three generalization tasks in the CALVIN simulation environment. Successfully completing these tasks require planning for and executing 3-7 actions.

Table 1: Generalization results in CALVIN. Mean success rates with STD from three seeds are reported. BLADE outperforms latent planning, LLM, and VLM baselines in completing novel long-horizon tasks.

Method	State Classifier	Latent Feasibility	Generalization Task		
			Abstract Goal	Geometric Constraint	Partial Observability
HULC [72]	N/A	N/A	2.78 ± 3.47	11.67 ± 11.55	0.00 ± 0.00
SayCan [6]	N/A	Short	23.89 ± 1.92	1.67 ± 2.89	1.67 ± 2.89
VILA [57]	N/A	N/A	18.38 ± 2.48	0.00 ± 0.00	4.17 ± 5.20
T2M-Shooting [51]	Learned	Long	57.78 ± 12.29	0.00 ± 0.00	13.33 ± 1.44
Ours	Learned	N/A	$\mathbf{68.33 \pm 10.14}$	$\mathbf{26.67 \pm 7.64}$	$\mathbf{75.83 \pm 3.82}$
T2M-Shooting [51]	GT	Long	61.67 ± 5.00	0.00 ± 0.00	0.83 ± 1.44
Ours	GT	N/A	$\mathbf{76.11 \pm 6.74}$	$\mathbf{56.67 \pm 16.07}$	$\mathbf{70.00 \pm 5.00}$

on simulator states, and there are in total 34 types of behaviors. We use RGB-D images from the mounted camera for classifier learning and partial 3D point clouds recovered from the images for policy learning. The original benchmark focuses only on evaluating individual skills and instruction following. To evaluate the ability to compositionally combine previously learned policies to solve novel tasks, we design six new generalization tasks, with examples shown in Fig. 4. Each task has a language instruction, a sampler that generates random initial states, and a goal satisfaction function for evaluation. For each task, we sample 20 initial states and evaluate all methods with three different random seeds. See Appendix B.1 for more details on the benchmark setup.

Baselines. We compare BLADE with two groups of baselines: hierarchical policies with planning in latent spaces and LLM/VLM-based methods for robotic planning. For the former, we use HULC [72], a representative method in CALVIN, which learns a hierarchical policy from language-annotated play data using hindsight labeling. For the latter, we use SayCan [6], Robot-VILA [57], and Text2Motion [51]. Note that Text2Motion assumes access to ground-truth symbolic states. Hence we compare Text2Motion with BLADE in two settings: one with the ground-truth states and the other with the state classifiers learned by BLADE. See Appendix B.2 for more details on these methods.

5.2 Results in Simulation

Table 1 presents the performance of different models in all three types of generalization tasks.

Structured behavior representations improve long-horizon planning. We first compare to the hierarchical policy HULC in Table. 1. BLADE with learned classifiers achieves a more than 65% improvement in the success rate for reaching abstract goals while using the same language-annotated play data. We attribute this to the particular implementation of hindsight labeling in HULC being not sufficient to generate plans that chain multiple high-level actions: for example, the task of placing all blocks in the closed drawer requires chaining together a minimum of 7 behaviors.

Structured transition models learned by BLADE facilitate long-horizon planning. Both SayCan and T2M-Shooting uses learned action feasibility models for planning. Shown in Table. 1, learning accurate feasibility models directly from raw demonstration data remains a significant challenge. In our experiment, we find that first, when the LLM does not take into account state information (SayCan), using the short-horizon feasibility model is not sufficient to produce sound plans. Second, since our model learns a structured transition model, factorized into different state predicates, BLADE is capable of producing more accurate longer-horizon plans than T2M-Shooting which learns long-horizon feasibility from data.

Structured scene representations facilitate making feasible plans. Compared to the Robot-VILA method, which directly predicts action sequences based on the image state, BLADE first uses learned state classifiers to construct an abstract state representation. This contributes to a 49% improvement on the Abstract Goal tasks in Table 1. We observe that the pre-trained VLM used in Robot-VILA often predicts actions that are not feasible in the current state. For example, Robot-VILA consistently performs better in completing "placing all blocks in a closed drawer" than "placing all blocks in an open drawer" since it always predicts opening the drawer as the first step.

Explicit modeling of geometric constraints and object visibility improves performance in these scenarios. BLADE can reason about these challenging situations without explicitly being trained in those settings. Table. 1 shows that our approach consistently outperforms baselines in these two settings. These generalization capabilities are built on the explicit modeling of geometric constraints and object visibility in behavior preconditions.

BLADE can automatically propose operators for the specific environment given demonstrations. Our experiment shows that the LLM can automatically propose high-quality behavior descriptions that resemble the dependency structures among operators. For example, the LLM discovers from the given contact primitive sequences and language-paired demonstration that blocks can only be placed after the block is lifted and that a drawer needs to be opened before placing objects inside, etc. Some of these dependencies are unique to the CALVIN environment, therefore requiring the LLM to generate specifically for this domain. We provide more visualizations in the Appendix A.1.

BLADE's automatic predicate annotation enables better classifier learning. From Table 1, we observe that having accurate state classifier models is critical for algorithms' performance (GT vs. Learned). Hence, we perform additional ablation studies on classifier learning. Prior work such as Migimatsu and Bohg [66] also presented a method for learning the preconditions and effects of actions from segmented trajectories and symbolic action descriptions. The key difference between BLADE and theirs is that they only use the first and last frame of each segment to supervise the learning of state classifiers. We compare the two classifier learning algorithms, given the same LLM-generated behavior definitions, by evaluating the classifier accuracy on held-out states. BLADE shows a 20.7% improvement in F1 (16.3% improvement for classifying object states and 38.6% improvement for classifying spatial relations) compared to the baseline model. This also translates into significant improvements in the planning success rate, as shown in Table 2.

Table 2: Ablation on state classifier learning in CALVIN.

Method	Abstract	Geometric	Partial Obs.
[66]	33.89 ± 5.85	9.17 ± 5.20	3.33 ± 2.89
BLADE	$\mathbf{68.33 \pm 10.14}$	$\mathbf{26.67 \pm 7.64}$	$\mathbf{75.83 \pm 3.82}$

5.3 Real World Experiments

Environments. We use a Franka Emika robot arm with a parallel jaw gripper. The setup includes five RealSense RGB-D cameras, with one being wrist-mounted on the robot and the remaining positioned around the workspace. Fig. 5 shows the two domains: Make Tea and Boil Water. For each domain, we collect 85 language-annotated demonstrations using teleoperation with a 3D mouse. After segmenting the demonstrations using proprioception sensor data, an LLM is used to generate behavior descriptions. These descriptions are subsequently used for policy and classifier learning.

Setup. We compare BLADE against the VLM-based baseline Robot-VILA. We omit SayCan and T2M-Shooting since they require additional training data. We first test the original action sequences seen in the demonstrations for each domain. We then test on tasks that require novel compositions of behaviors for four types of generalizations, i.e., unseen initial condition, state perturbation, geometric constraints, and partial observability. For each generalization type, we run six experiments and report the number of experiments that have been successfully completed. See Appendix D for details.

Results. In Fig. 5, we show that our model is able to successfully complete at least 4/6 tasks for all generalization types in the two different domains. In comparison, Robot-VILA struggles to generate

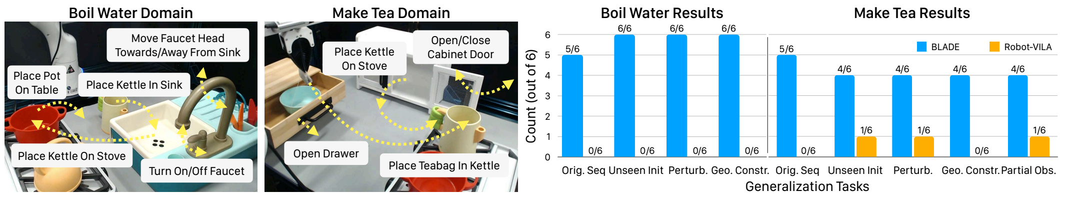


Figure 5: Domains and Results in Real World. Make Tea features a toy kitchen designed to simulate boiling water on a stove. The robot must assess the available space on the stove for the kettle. It also needs to manage the dependencies between actions, such as the faucet must be turned away before the kettle can be placed into the sink to avoid collisions. **Boil Water** involves a tabletop task aimed at preparing tea, incorporating a cabinet, a drawer, and a stove. The robot must locate the kettle, potentially hidden within the cabinet, and a teabag in the drawer. Additionally, it must consider geometric constraints by removing obstacles that block the cabinet doors. In both environments, our model significantly outperforms the VLM-based planner Robot-VILA.

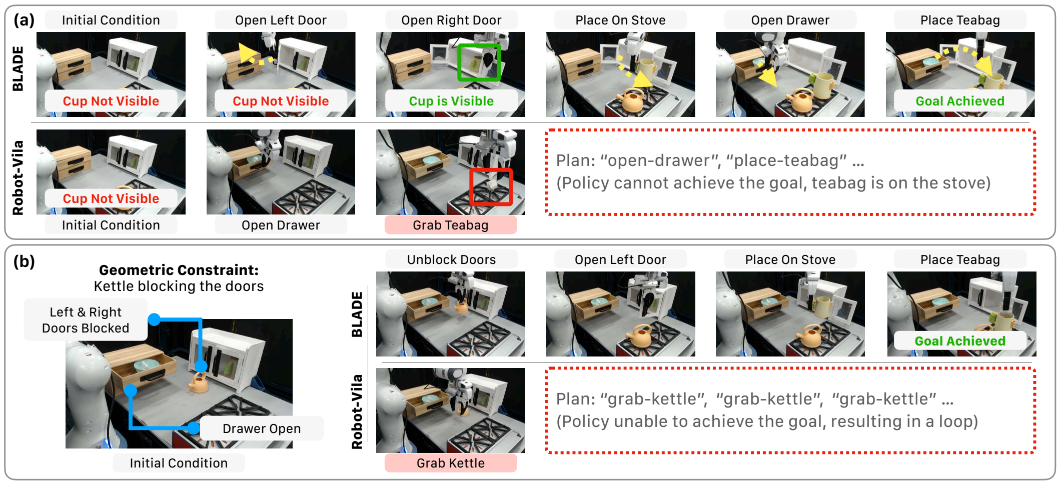


Figure 6: Real World Planning and Execution. We show the execution traces from BLADE and Robot-VILA for two generalization tasks: (a) partial observability and (b) geometric constraints.

correct plans to complete the tasks. In Fig. 6, we visualize the generated plans and execution traces of both models. In example (a), we show that BLADE can find the kettle initially hidden in the cabinet and then complete the rest of the task. In comparison, Robot-VILA directly predicts placing the teabag in the kettle when the kettle is not visible, resulting in a failure.

6 Conclusion and Discussion

BLADE is a novel framework for long-horizon manipulation by integrating model-based planning and imitation learning. BLADE uses an LLM to generate behavior descriptions with preconditions and effects from language-annotated demonstrations and automatically generates state abstraction labels based on behavior descriptions for learning state classifiers. At performance time, BLADE generalizes to novel states and goals by composing learned behaviors with a planner. Compared to latent-space and LLM/VLM-based planners, BLADE successfully completes significantly more long-horizon tasks with various types of generalizations.

Limitations. One limitation of BLADE is that the automatic segmentation of demonstrations is based on gripper states; more advanced contact detection techniques might be required for certain tasks such as caging grasps. We also assume the knowledge of a given set of predicate names in natural language and focus on learning dependencies between actions using the given predicates. Automatically inventing task-specific predicates from demonstrations and language annotations, possibly with the integration of vision-language models (VLMs) is an important future direction. In our experiments, we also found that noisy state classification led to some planning failures. Therefore, developing planners that are more robust to noises in state estimation is necessary. Finally, achieving novel compositions of behaviors also requires policies with strong generalization to novel environmental states, which remain a challenge for skills learned from a limited amount of demonstration data.

Acknowledgments

This work is in part supported by Analog Devices, MIT Quest for Intelligence, MIT-IBM Watson AI Lab, ONR Science of AI, NSF grant 2214177, ONR N00014-23-1-2355, AFOSR YIP FA9550-23-1-0127, AFOSR grant FA9550-22-1-0249, ONR MURI N00014-22-1-2740, ARO grant W911NF-23-1-0034. We extend our gratitude to Jonathan Yedidia, Nicholas Moran, Zhutian Yang, Manling Li, Joy Hsu, Stephen Tian, Chen Wang, Wenlong Wang, Yunfan Jiang, Chengshu Li, Josiah Wong, Mengdi Xu, Sanjana Srivastava, Yunong Liu, Tianyuan Dai, Wensi Ai, Yihe Tang, the members of the Stanford Vision and Learning Lab, and the anonymous reviewers for insightful discussions.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023. 1, 5, 18, 19

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. In *ICAPS*, 2020. 1, 2

[3] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *ICRA*, 2021.

[4] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *IJRR*, 43(4):533–549, 2024. 1

[5] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *CoRL*, 2020. 1

[6] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *CoRL*, 2023. 1, 2, 6

[7] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer. In *ICRA*, 2014. 2

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *RSS*, 2016.

[9] M. Toussaint. Logic-Geometric Programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, 2015.

[10] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *ICRA*, 2022.

[11] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *ICRA*, 2020.

[12] Z. Yang, C. R. Garrett, T. Lozano-Pérez, L. Kaelbling, and D. Fox. Sequence-based plan feasibility prediction for efficient task and motion planning. In *RSS*, 2023. 2

[13] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017. 2

[14] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *ICLR*, 2020.

[15] H. Shi, H. Xu, S. Clarke, Y. Li, and J. Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. In *CoRL*, 2023.

[16] A. Simeonov, Y. Du, B. Kim, F. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez. A long horizon planning framework for manipulating rigid pointcloud objects. In *CoRL*, 2021.

[17] X. Lin, C. Qi, Y. Zhang, Z. Huang, K. Fragkiadaki, Y. Li, C. Gan, and D. Held. Planning with spatial and temporal abstraction from point clouds for deformable object manipulation. In *CoRL*, 2022.

[18] Y. Du, M. Yang, P. Florence, F. Xia, A. Wahid, B. Ichter, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum, et al. Video language planning. *arXiv:2310.10625*, 2023. 2

[19] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine. Multi-stage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 2024. 2

[20] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv:2403.12910*, 2024.

[21] S. Pirk, K. Hausman, A. Toshev, and M. Khansari. Modeling long-horizon tasks as sequential interaction landscapes. In *CoRL*, 2020.

[22] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *CoRL*, 2023.

[23] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021. 2

[24] O. Mees, J. Borja-Diaz, and W. Burgard. Grounding language with visual affordances over unstructured data. In *ICRA*, 2023. 21

[25] A. Mandlekar, C. Garrett, D. Xu, and D. Fox. Human-in-the-loop task and motion planning for imitation learning. In *CoRL*, 2023. 2

[26] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs. Universal Visual Decomposer: Long-horizon manipulation made easy. In *ICRA*, 2024. 2

[27] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022. 2

[28] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *ICRA*, 2020. 2

[29] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *CVPR*, 2019.

[30] D.-A. Huang, D. Xu, Y. Zhu, A. Garg, S. Savarese, F.-F. Li, and J. C. Niebles. Continuous relaxation of symbolic planner for one-shot imitation learning. In *IROS*, 2019.

[31] Y. Huang, N. C. Taylor, A. Conkey, W. Liu, and T. Hermans. Latent space planning for multi-object manipulation with environment-aware relational classifiers. *IEEE Transactions on Robotics*, 2024. 2

[32] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289, 2018. 2

[33] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum. Predicate invention for bilevel planning. In *AAAI*, 2023.

[34] A. Ahmetoglu, E. Oztop, and E. Ugur. Symbolic manipulation planning with discovered object and relational predicates. *arXiv preprint arXiv:2401.01123*, 2024.

[35] N. Shah, J. Nagpal, P. Verma, and S. Srivastava. From reals to logic and back: Inventing symbolic vocabularies, actions and models for planning from raw data. *arXiv preprint arXiv:2402.11871*, 2024.

[36] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu. Interpret: Interactive predicate learning from language feedback for generalizable task planning. In *RSS*, 2024. 2

[37] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*, 2022. 2

[38] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*, 2022. 2

[39] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati. On the planning abilities of large language models-a critical investigation. In *NeurIPS*, 2023. 2

[40] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv:2402.01817*, 2024. 2

[41] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. AutoTAMP: Autoregressive task and motion planning with llms as translators and checkers. In *ICRA*, 2024. 2

[42] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv:2304.11477*, 2023.

[43] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *arXiv:2302.05128*, 2023.

[44] A. Mavrogiannis, C. Mavrogiannis, and Y. Aloimonos. Cook2ltl: Translating cooking recipes to ltl formulae using large language models. In *ICRA*, 2024. 2

[45] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz. Generalized planning in PDDL domains with pretrained large language models. In *AAAI*, 2024. 2

[46] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, et al. Ghost in the Minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv:2305.17144*, 2023. 2

[47] K. Nottingham, P. Ammanabrolu, A. Suhr, Y. Choi, H. Hajishirzi, S. Singh, and R. Fox. Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling. In *ICML*, 2023. 2

[48] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu. Reasoning with language model is planning with world model. In *EMNLP*, 2023. 2

[49] 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. In *ICRA*, 2023. 2

[50] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *ICRA*, 2023. 2

[51] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023. 2, 6

[52] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg. Replan: Robotic replanning with perception and language models. *arXiv:2401.04157*, 2024. 2

[53] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv:2303.03378*, 2023. 2

[54] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan. Embodied task planning with large language models. *arXiv:2307.01848*, 2023.

[55] J. Xiang, T. Tao, Y. Gu, T. Shu, Z. Wang, Z. Yang, and Z. Hu. Language models meet world models: Embodied experiences enhance language models. In *NeurIPS*, 2024. 2

[56] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K. Y. Chen, N. Chin, P. Dan, X. Fan, et al. Mosaic: A modular system for assistive and interactive cooking. *arXiv preprint arXiv:2402.18796*, 2024. 2

[57] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of GPT-4v in robotic vision-language planning. *arXiv:2311.17842*, 2023. 2, 6

[58] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi. ChatGPT empowered long-step robot control in various environments: A case application. *IEEE Access*, 2023. 2

[59] L. Wong, J. Mao, P. Sharma, Z. S. Siegel, J. Feng, N. Korneev, J. B. Tenenbaum, and J. Andreas. Learning adaptive planning representations with natural language guidance. In *ICLR*, 2024. 2

[60] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In *NeurIPS*, 2023.

[61] P. Smirnov, F. Joublin, A. Ceravola, and M. Gienger. Generating consistent PDDL domains with large language models. *arXiv:2404.07751*, 2024. 2

[62] Z. Li, K. Yu, S. Cheng, and D. Xu. League++: Empowering continual robot learning through guided skill acquisition with large language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. 2

[63] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. In *ICLR*, 2024. 2

[64] V. Lifschitz. On the semantics of STRIPS. In M. Georgeff, Lansky, and Amy, editors, *Reasoning about Actions and Plans*, pages 1–9. Morgan Kaufmann, San Mateo, CA, 1987. 3

[65] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, and D. Christianson. PDDL: The Planning Domain Definition Language, 1998. 4

[66] T. Migimatsu and J. Bohg. Grounding predicates through actions. In *ICRA*, 2022. 5, 7

[67] J. Mao, T. Lozano-Pérez, J. Tenenbaum, and L. Kaelbling. PDSketch: Integrated domain programming, learning, and planning. In *NeurIPS*, 2022. 5

[68] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding Dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv:2303.05499*, 2023. 5, 17

[69] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001. 5

[70] M. Helmert. The fast downward planning system. *JAIR*, 26:191–246, 2006. 5

[71] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *RA-L*, 7:7327–7334, 2021. 5

[72] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *RA-L*, 7:11205–11212, 2022. 6, 21

[73] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, et al. Recognize Anything: A strong image tagging model. In *CVPR*, 2024. 14

[74] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2023. 16

[75] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3D Diffuser Actor: Policy diffusion with 3D scene representations. *arXiv:2402.10885*, 2024. 16

[76] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs. Universal Visual Decomposer: Long-horizon manipulation made easy. In *ICRA*, 2024. 16

[77] W. Wan, Y. Zhu, R. Shah, and Y. Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *ICRA*, 2024. 16

[78] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 17

[79] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *ICRA*, 2011. 19, 20

[80] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox. Online replanning in belief space for partially observable task and motion problems. In *ICRA*, 2020. 20