Grounding Language Plans in Demonstrations Through Counterfactual Perturbations

Yanwei Wang, Tsun-Hsuan Wang, Jiayuan Mao, Michael Hagenow, Julie Shah
MIT CSAIL

***Abstract*—Grounding the common-sense reasoning of Large Language Models (LLMs) in physical domains remains a pivotal yet unsolved problem for embodied AI. Whereas prior works have focused on leveraging LLMs directly for planning in symbolic spaces, this work uses LLMs to guide the search of task structures and constraints implicit in multi-step demonstrations. Specifically, we borrow from manipulation planning literature the concept of mode families, which group robot configurations by specific motion constraints, to serve as an abstraction layer between the high-level language representations of an LLM and the low-level physical trajectories of a robot. By replaying a few human demonstrations with synthetic perturbations, we generate coverage over the demonstrations' state space with additional successful executions as well as counterfactuals that fail the task. Our explanation-based learning framework trains an end-to-end differentiable neural network to predict successful trajectories from failures and as a by-product learns classifiers that ground low-level states and images in mode families without dense labeling. The learned grounding classifiers can further be used to translate language plans into reactive policies in the physical domain in an interpretable manner. We show our approach improves the interpretability and reactivity of imitation learning through 2D navigation and simulated and real robot manipulation tasks. Website: https://yanweiw.github.io/glide/**

I. Introduction

Language models, in particular, pretrained large language models (LLMs) contain a large amount of knowledge about physical interactions in an abstract space. However, a grand open challenge lies in extracting such semantic knowledge and grounding it in physical domains to solve multi-step tasks with embodied agents. Previous methods, given the symbolic and abstract nature of language, primarily focus on leveraging LLMs to propose abstract actions or policies in purely symbolic spaces or on top of manually defined high-level primitive abstractions [23, 1, 35]. Such approaches inherently require a set of predefined primitive skills and additional toolkits for estimating affordances or feasibility before executing a plan generated by an LLM [1, 22].

To address this important limitation, in this paper, we consider the problem of grounding plans in abstract language spaces into robot demonstration trajectories, which lie in the low-level robot configuration spaces. Our key idea is that many verbs; such as reach, grasp, and transport; are all grounded on top of mode families that are lower-dimensional manifolds in the configuration space [as in manipulation mechanics, see 27, 14]. Therefore, LLMs can be prompted to describe the multi-step structure of demonstrations in terms of semantic mode abstractions: valid mode transitions describe pre-conditions for mode-based skills, and mode boundaries explicitly encode motion constraints in the physical space that are critical for task success.

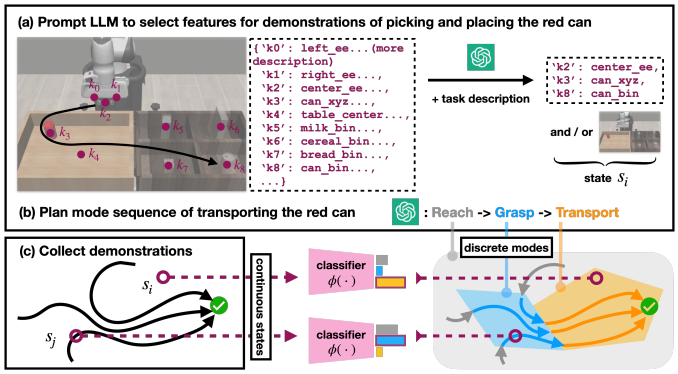

Fig. 1: **GLiDE framework** Given a common-sense LLM that understands (a) the appropriate state abstractions for a task and (b) how to solve the task via a sequence of manipulation modes in semantic space and (c) a few unsegmented human demonstrations that embody the transitions through these modes, we learn a grounding classifier that maps continuous physical states and observations to discrete semantic modes. Mode boundaries discovered by the classifier encode constraints implicit in the demonstrations that are critical for task success.

Building upon this idea, we propose ***G**rounding **L**anguage in **DE**monstrations* (GLiDE, illustrated in Fig. 1), which casts the language grounding problem into two stages: learning to classify current modes from states, and learning mode-specific policies. The main challenge in mode classification is that learning a decision boundary fundamentally requires both positive and negative labeled examples. To avoid having humans exhaustively provide dense mode annotation that covers the entire state space, we propose to systematically perturb demonstrations to generate "counterfactual" trajectories and use a simple "overall" task success predictor as sparse supervision. Intuitively, perturbations to inconsequential parts of a successful replay add unseen state coverage, while perturbations that cause counterfactual failing outcomes reveal constraints in the demonstration. Next, we use an explanation-based learning paradigm [8, 29] to recover the mode families that successful demonstrations implicitly transition through. With a learned classifier that maps continuous physical states to discrete abstract modes, we can then learn mode-specific policies and also use LLMs to plan for recovery from external perturbations or other sources of partial failures. Our system improves both the interpretability and reactivity of robot learning of multi-step tasks.

Our framework of grounding language plans as recovering modes and learning mode-specific policies brings two important

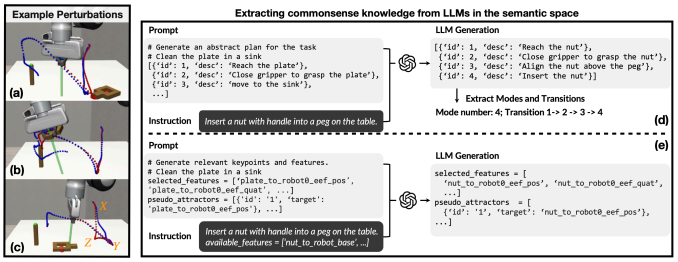

Fig. 2: (a-c) Example perturbations causing replays (blue) to deviate from successful demonstrations (red). The task is to pick up the square nut and place it on the peg. End-effector perturbations at different locations (a) may or (b) may not cause grasp failures. (c) The gripper picks up the nut despite an initial end-effector perturbation but later drops it due to a gripper perturbation. LLMs can be prompted (d) to describe a task solution via a discrete mode sequence or (e) to select relevant features and pseudo attractors.

advantages. First, compared to frameworks that generate robot behavior solely based on text, we do not require pre-built policies and feasibility predictors for primitive actions. Experiments show that our learning paradigm can successfully identify each mode from the demonstration data without any human segmentation annotations, and from only a small number of expert-generated demonstrations. Second, connecting demonstrations with language suggests a principled way to improve the interpretability and reactivity of motion imitation. While plenty of data collection systems [39, 10, 38, 11, 6] allow humans to demonstrate complex multi-step tasks, these demonstrations are typically unsegmented without semantic annotations of individual steps. Neither do humans elaborate on the task constraints that successful trajectories implicitly satisfy. Consequently, the resulting imitation policies cannot detect whether current actions fail to achieve pre-conditions [13] of subsequent actions or replan to recover from mistakes due to covariate shift [28]. Our system enables the usage of LLMs for replanning and improves the overall system robustness.

II. Method

Our framework, GLiDE, takes in a language description of the target task, and a small set of successful human demonstrations as input, and aims to produce a robust policy that can accomplish the task successfully even under perturbations. GLiDE first uses a perturbation strategy to augment a small set of human demonstrations with additional successful executions and failing counterfactuals. (Section II-A). Next, it prompts a large language model (LLM) to decompose the very high-level instruction into a step-by-step abstract plan in language. At this step, the most important outcome is a feasibility matrix that encodes how we can transition between different modes in this task (Section II-B). Given the augmented demonstration and perturbation dataset and the LLM-generated abstract plan, we ground each mode onto trajectories (Section II-C) and generate motions for individual modes to be sequenced by a language plan (Section II-D).

A. Demonstration Data Augmentation with Counterfactual Perturbations

To learn a grounding classifier that can partition the state space being considered into mode families, we need data coverage beyond the regions explored in a few successful demonstrations. Additionally, to learn mode abstractions that can be used to predict task success–as opposed to clustering data based on statistical similarity–negative data that fail by crossing infeasible boundaries are necessary. Assuming an oracle that can label the execution outcome of a synthetically generated trajectory, we propose the following perturbations to demonstration replays that might reveal task constraints:

End-effector perturbations Illustrated in Fig. 2c, given a successful demonstration shown in blue, we first sample two points on the trajectory, namely X and Y. Next, we randomly sample a third point Z in the state space. During the replay shown in red, we replace the XY segment with XZ and ZY. Depending on the location and magnitude of the perturbations, the robot may still succeed in the task (Fig. 2b) or fail (Fig. 2a), revealing that grasping the square nut is a pre-condition for the next step of peg insertion to be successful.

Gripper perturbations Illustrated in Fig. 2c, we randomly toggle the gripper state while otherwise adhering to the original trajectory. Failure replays where the gripper drops the nut prematurely reveal the motion constraint of holding the nut during transportation.

Given the perturbed trajectories, we execute them using a trajectory-following controller in the environment and collect a binary task success signal for each trajectory. Essentially, this gives us a dataset of paired trajectories and their task success labels: $\langle \tau^i, \mathit{succ}^i \rangle$, where $\tau^i = \{s_1, s_2, \cdots, s_T\}$, and $\mathit{succ}^i \in \{0, 1\}$. To learn the grounding classifier $\phi(\cdot)$ that can map τ^i to its corresponding mode sequence $\phi(\tau^i) = \{m_1, m_2, \cdots, m_T\}$ (mode and mode families are used interchangeably in this work), we ask LLMs what modes there are in a demonstration, how they are connected, and what constitutes a state s_t for a given task.

B. Semantic description of demonstrations and task structure from LLMs

Explaining continuous demonstrations with a discrete mode sequence First, we assume a given small set of demonstrations $\{\langle \xi^i, 1 \rangle\}_{i=1}^{D}$, which can be variable at the motion level, satisfy the same sequential transition through K modes, defined as $\sigma \in \Sigma = \{\sigma_i\}_{i=1}^{K}$ and $\phi(s_t) = m_t \in \Sigma$. In other words, if we reduce self-transitions in the demonstrations where $m_t = m_{t+1}$, mode sequence $\phi(\xi^i)$ for all demonstrations can be reduced to the same K-step transitions $\sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \sigma_K$. This is the form of the language plan we prompt LLMs to generate to describe demonstrations. The plan informs the number of modes there are as well as the semantic grounding of each mode as seen in Fig. 2d.

Representing states with task-informed abstraction Second, we further prompt LLMs to define the state representation s_t as a set of keypoint-based features or image observations that are relevant to mode classification. In particular, the keypoint-based features come from a pre-defined exhaustive list of keypoints describing the scene as seen in Fig. 1a. Each keypoint definition contains (1) the keypoint name and (2) a short description of its semantic meaning. Given a task description, an LLM can

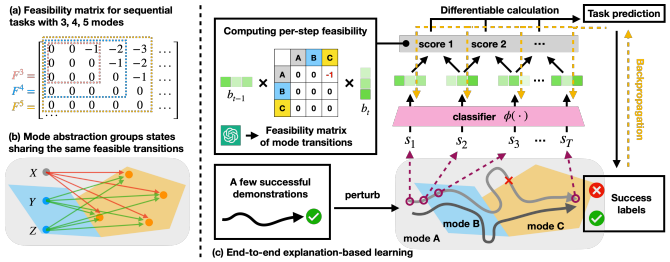

Fig. 3: (a) Example feasibility matrices. Specifically, F^3 can describe the modal structure for a pick-and-place task with solution reach→grasp→transport, where reach→transport directly is infeasible. (b) The definition of a mode transition implies every state in the second mode is reachable from every state in the first mode (states Y and Z are in the same mode but not X). We leverage this connection between the continuous states and the discrete modes to design (c) a fully-differentiable pipeline that calculates overall trajectory success based on the mode classification of individual states in the trajectory.

be prompted to either select a subset of keypoints tracking absolute locations or combine pairs of keypoints to track relative positions as shown in Fig. 2e. For image observations, we either use the raw image as a state representation or use a pre-trained vision model [19] or vision-language model [16] to extract features from the image.

Encoding discrete modal structure in a feasibility matrix Lastly, while successful demonstrations ξ^i can be reduced to a K-step language plan, not every perturbed trajectory τ^i can be as it might not be successful or correspond to a minimal solution. Therefore, the reduced mode sequence may contain back-and-forth steps such as $\sigma_1 \to \sigma_2 \to \sigma_1 \to \cdots$ or simply invalid mode transitions. To describe the modal structure of a task in terms of the feasible transitions between modes, we generate a feasibility matrix F^K with K modes by first querying LLMs whether two semantic modes are directly connected. Then we compute the matrix entry F_{ij} from LLM responses as the negative shortest path between each pair of modes. In the case of sequential tasks with a linear temporal structure (true for most experiments considered in this work), zero entries F_{ij} encode valid transitions that incur zero costs. Negative entries F_{ij} encode infeasible transitions, and the magnitudes denote the number of missing modes in between. In particular, in Fig. 3a diagonal entries F_{ii} are feasible self-transitions, and entries $F_{i,i+1}$ are demonstrated mode transitions towards the goal. Note for tasks with complex structures, the matrix may have more negative entries than the ones shown in Fig. 3a. The feasibility matrix is also interpretable and can be modified manually by humans.

C. *End-to-end explanation-based learning for mode classification*

Given a language plan, a task-informed state representation, and a feasibility matrix as discrete structural information about a task, learning the grounding classifier is an inverse problem that tries to recover the underlying modal structure from sparsely labeled continuous trajectories. To this end, we design a differentiable decision-making pipeline to explain the task success of a trajectory on top of mode predictions. Having trajectory coverage with contrasting execution outcomes allows for recovering the precise grounding in terms of mode boundaries.

Mode classifier Our mode classifier is a neural network (with softmax output layers) that inputs a state s_t and outputs a categorical distribution of the abstract mode at that state. Overloading the notation $\phi(\cdot)$ to output both a predicted mode m_t and a mode belief, we have $b_t = \phi(s_t)$. The architecture of the classifier depends on the state representation. The number of softmax categories K is chosen based on the sequence length of the LLM-generated plan. If we had dense mode annotations $\{\langle s_t, m_t \rangle\}_{t=1}^T$, we could train the classifier directly with a cross-entropy loss. However, we only have supervision at the trajectory level via task success. Therefore, we need a differentiable forward model that can predict task success from a sequence of mode beliefs $\{b_t\}_{t=1}^T$.

Differentiable forward model to predict task success What makes a perturbed trajectory rollout unsuccessful (or still successful) in solving a task? Following the approach by Wang et al. [36], we consider a successful trajectory as one that both (1) contains only feasible mode transitions according to F^K and (2) eventually reaches the final mode σ_K seen in the demonstrations. Since the perturbations we consider in this work do not affect the starting state s_1 and final state s_T, the success criteria for a trajectory τ^+ solely concerns intermediate transitions: $\phi(s_t^+)F^K\phi(s_{t+1}^+) = 0, \forall s_t^+ \in \tau^+$. Similarly, a failure trajectory τ^- is one that contains at least one invalid mode transition.

To operationalize this idea, let's consider the dataset of trajectories $\mathcal{T}$ containing both successful trajectories $\mathcal{T}^+ = \{\tau^{i+}\}_{i=1}^M$ and failure trajectories $\mathcal{T}^- = \{\tau^{j-}\}_{j=1}^N$. First, we use a cross-entropy loss to enforce that the starting and ending continuous states for all trajectories must be in the initial and final mode being demonstrated: $\mathcal{L}_{init} = \mathbb{E}_{\tau_i \sim \mathcal{T}} \mathcal{L}_{CE}(\phi(s_1^i), \sigma_1)$ and $\mathcal{L}_{final} = \mathbb{E}_{\tau_i \sim \mathcal{T}} \mathcal{L}_{CE}(\phi(s_T^i), \sigma_K)$. Second, we define the success and failure loss using $f_{t,t+1}$, which is a shorthand for transition feasibility score $\phi(s_t)F^K\phi(s_{t+1})$ between two states:

$$\begin{aligned} \mathcal{L}_{succ} &= -\frac{1}{M} \sum_{\tau_i \in \mathcal{T}^+} \frac{1}{T-1} \sum_{t=1}^{T-1} f_{t,t+1} \\ \mathcal{L}_{fail} &= \frac{1}{N} \sum_{\tau_j \in \mathcal{T}^-} \max(-1, \sum_{t=1}^{T-1} f_{t,t+1}) \end{aligned} \tag{1}$$

Intuitively, minimizing $\mathcal{L}_{succ}$ encourages the classifier to predict mode beliefs such that all transitions between consecutive states are feasible. Minimizing $\mathcal{L}_{fail}$ encourages the classifier to predict mode beliefs such that there exists at least one invalid mode transition. The clipping in $\mathcal{L}_{fail}$ at -1 makes the loss well-defined and treats all invalid mode transitions described by the negative entries in Fig. 3a equally*.

Fig. 3b gives another intuitive example, where states Y and Z constitute the same mode but not state X. A necessary

*Empirically, setting all negative entries in the matrix to be -1 can get gradient descent optimization stuck.

condition to test if a state s is in mode σ_i is to check if s can directly transition to at least one state in mode σ_{i+1} in the trajectory. Adding everything together, we have $\mathcal{L}_{full}$ in Eq. 2, where λ_s, λ_f, and λ_i are hyperparameters for balancing loss terms:

$$\mathcal{L}_{full} = \lambda_s \mathcal{L}_{succ} + \lambda_f \mathcal{L}_{fail} + \lambda_i (\mathcal{L}_{init} + \mathcal{L}_{final}) \tag{2}$$

Extension to underactuated systems. These conditions are sufficient for recovering modes from a fully-actuated system. For underactuated systems [32] such as object manipulation where objects cannot directly move from one configuration to another via teleportation, it is not possible to generate a direct transition between any two modes such as the ones shown in 3b using synthetic perturbations. Hence, we need an additional regularization at the motion level to infer precise boundaries. Specifically, states in the same mode should go through similar dynamics. In other words, one should be able to infer $(s_{t+1} - s_t)$ from $(s_t - s_{t-1})$. Such mapping should be different for different modes. For example, the relative transformation between the end-effector pose and the object pose should remain the same when the robot is rigidly holding the object and change otherwise. Based on this observation, we instantiate a forward dynamics model $\psi(\cdot)$ that inputs the current state change and predicts how the state should change next for each mode. Coupled with a mode belief, we can predict the next state change as $\phi(s_t)^\intercal \psi(s_t - s_{t-1})$. Consequently, we can train mode classifiers for underactuated systems by introducing a dynamics loss $\mathcal{L}_{dyn}$:

$$\mathcal{L}_{under} = \mathcal{L}_{full} + \lambda_d \mathcal{L}_{dyn} \quad \text{where}$$
$$\mathcal{L}_{dyn} = \sum_{\tau_j \in \mathcal{T}} \sum_{t=1}^{T-1} \|\phi(s_t)^\intercal \psi(s_t - s_{t-1}) - (s_{t+1} - s_t)\|_2^2 \tag{3}$$

Minimizing $\mathcal{L}_{dyn}$ groups states into modes based on similarity in dynamics. Since losses are differentiable with respect to ϕ and ψ, we use stochastic gradient descent to optimize learnable parameters.

D. Mode-Based Motion Generation

Having learned the explicit mode boundaries, we can leverage them in motion planning to ensure that the robot avoids invalid mode transitions [20]. Alternatively, we can use the classifier $\phi(\cdot)$ to segment demonstrations into mode-specific datasets, with which we can learn imitation policies $\pi_k(a|s)$ for each mode σ_k and sequence them using a discrete plan [36]. To further improve the robustness of the learned policy for manipulation tasks, we use the mode feature identified by the LLM to construct a pseudo-attractor for each mode. If the mode feature is the absolute pose of the robot end-effector, we compute the mean end-effector poses at which mode transitions $\sigma_k \to \sigma_{k+1}$ occur as the pseudo-attractor; if it is a relative pose, we transform that into an absolute pose of the end-effector at test time. We use this pseudo-attractor to construct a potential field that guides the robot to move towards the next mode at inference time. Specifically, the final mode-based policy $\pi_k^*(a|s)$ is a weighted sum of the original $\pi_k(a|s)$

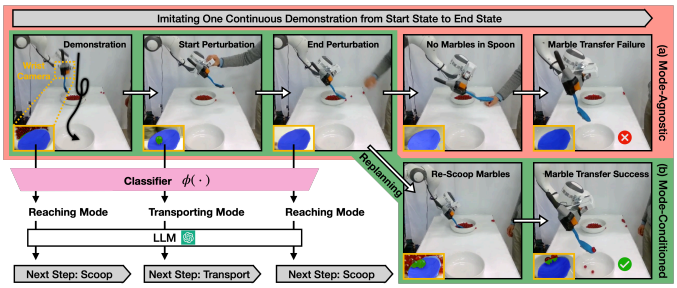

Fig. 4: Illustration of the robot scooping task. Here the perturbations are human-initiated (e.g., moving the spoon to drop or fail to grasp marbles). A mode-agnostic BC that imitates continuous motion demonstrations cannot replan at the task level when all marbles are dropped during the transporting mode (a). In contrast, our mode-conditioned policy enabled by the grounding classifier can leverage LLMs to replan given external perturbations (b).

and a control command that moves the end-effector towards the pseudo-attractor for mode σ_k. We only apply the pseudo-attractor term when the distance between the current state and the pseudo-attractor is greater than a threshold. Intuitively, when a large perturbation leads to out-of-distribution states, the potential field will drive the system back to the demonstration distribution before the imitation policy π_k takes sole effect.

III. Experiments

Marble scooping task To show the learned mode abstractions facilitate symbolic planning to recover from mistakes, we consider a scooping task where a robot needs to scoop and transport at least one marble from one bowl to another despite arbitrary physical perturbations. Existing approach such as temporal logic imitation [36] requires engineering a marble detector to check if the spoon is holding marbles to plan actions accordingly. Instead, we learn a marble classifier from contrasting successful and failing demonstrations. Specifically, we record the robot end-effector's poses and wrist camera views as a human demonstrates both successful scooping from various starting states and various failure executions through kinesthetic teaching. To improve learning efficiency, we preprocess the raw wrist image to extract a mask corresponding to marble objects, where an empty spoon returns an empty mask. The classifier is then trained on a state representation consisting of end-effector poses and pixels inputs to predict a semantic mode. We show in Fig. 7 a mode-agnostic imitation policy cannot recover from task-level perturbations (dropping marbles) while a mode-based policy can leverage LLM to replan to ensure successful execution. For additional experiment results, please refer to the appendix or the website.

IV. Conclusion

In conclusion, this work introduces a framework, ***G**rounding **L**anguage in **DE**monstrations* (GLiDE), to effectively ground the knowledge within large language models into physical domains, via mode families. Given a small number of human demonstrations and task descriptions, we show how GLiDE successfully recovers mode families and their transitions required in the task and enables the learning of robust robot control policies.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I Can, Not as I Say: Grounding Language in Robotic Affordances. *arXiv:2204.01691*, 2022.

[2] Jacob Andreas and Dan Klein. Alignment-Based Compositional Semantics for Instruction Following. In *EMNLP*, 2015.

[3] Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. In *ICML*, 2017.

[4] Christopher Bradley, Adam Pacheck, Gregory J. Stein, Sebastian Castro, Hadas Kress-Gazit, and Nicholas Roy. Learning and planning for temporally extended tasks in unknown environments, 2021.

[5] Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, pages 6276–6282, 2019.

[6] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[7] Rodolfo Corona, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell. Modular Networks for Compositional Instruction Following. In *NAACL-HLT*, 2021.

[8] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine learning*, 1:145–176, 1986.

[9] Eoin Delaney, Derek Greene, and Mark T Keane. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pages 32–47. Springer, 2021.

[10] Hongjie Fang, Hao-Shu Fang, Yiming Wang, Jieji Ren, Jingjing Chen, Ruo Zhang, Weiming Wang, and Cewu Lu. Low-cost exoskeletons for learning whole-arm manipulation in the wild. *arXiv preprint arXiv:2309.14975*, 2023.

[11] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[12] Divyansh Garg, Skanda Vaidyanath, Kuno Kim, Jiaming Song, and Stefano Ermon. Lisa: Learning interpretable skill abstractions from language. *Advances in Neural Information Processing Systems*, 35:21711–21724, 2022.

[13] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated Task and Motion Planning. *Ann. Rev. Control Robot. Auton. Syst.*, 4:265–293, 2021.

[14] Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915, 2010.

[15] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *ICML*, 2022.

[16] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv:2307.05973*, 2023.

[17] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an Abstraction for Hierarchical Deep Reinforcement Learning. In *NeurIPS*, 2019.

[18] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[20] Steven LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Research Report 9811*, 1998.

[21] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[22] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

[23] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+ P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv:2304.11477*, 2023.

[24] Zhezheng Luo, Jiayuan Mao, Jiajun Wu, Tomás Lozano-Pérez, Joshua B Tenenbaum, and Leslie Pack Kaelbling. Learning Rational Subgoals from Demonstrations and Instructions. In *AAAI*, 2023.

[25] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[26] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[27] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001.

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[29] Alberto Segre and Gerald DeJong. Explanation-based ma-

nipulator learning: Acquisition of planning ability through observation. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 555–560. IEEE, 1985.

[30] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill Induction and Planning with Latent Language. In *ACL*, 2022.

[31] Shao-Hua Sun, Te-Lin Wu, and Joseph J Lim. Program guided agent. In *ICLR*, 2020.

[32] Russ Tedrake. *Underactuated Robotics*. 2023. URL https://underactuated.csail.mit.edu.

[33] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *AAAI*, 2011.

[34] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Teaching multiple tasks to an rl agent using ltl. In *AAMAS*, 2018.

[35] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv:2305.16291*, 2023.

[36] Yanwei Wang, Nadia Figueroa, Shen Li, Ankit Shah, and Julie Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. *arXiv preprint arXiv:2206.04632*, 2022.

[37] Yanwei Wang, Ching-Yun Ko, and Pulkit Agrawal. Visual pre-training for navigation: What can we learn from noise? *arXiv preprint arXiv:2207.00052*, 2022.

[38] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

[39] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[40] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX

We evaluate our method on three sets of experiments: (1) a 2D navigation task, (2) simulated robot manipulation tasks in RoboSuite [40], and (3) a real-robot implementation of the 2D navigation and a marble-scooping task. Since our robot experiments use end-effector pose control, we refer to task-space features as states rather than configurations.

A. 2D Navigation

Setup The 2D navigation environment consists of a sequence of connected randomly generated polygons, and the goal is to traverse from any state in the free space (mode 1) through the polygon sequence consecutively as demonstrated until reaching the final polygon. This environment serves as a 2D abstraction of the modal structure for multi-step manipulation tasks, where each polygon represents a different mode with its boundaries showing the constraint of the mode. Illegal transitions include non-consecutive jumps between modes such as direct transitions from free space to any later modes other than mode 2. This system is fully-actuated with (x, y) coordinate as the state representation and $(\dot{x}, \dot{y})$ as the agent action. For all environments, we use fewer than 10 successful demonstrations for classifier learning and policy learning.

Results: Mode classification We visualize the learned grounding classifier in Fig. 5b and Appendix C. Compared to baselines in Fig. 5(c-e), the mode boundaries recovered by GLiDE are the closest to the ground truth shown in Fig. 5a. In particular, the poor grounding learned in Fig. 5(c-e) shows respectively the importance of learning with counterfactual data, a correct task specification from LLMs, and a task prediction loss beyond clustering solely based on statistical similarities in the data. Quantitative results and more visualizations can be found on our website.

Results: Task execution Next, we show the learned grounding classifier can be used to improve task success rates, especially in the face of external perturbations. We use behavior cloning (BC) as a mode-agnostic baseline to learn a single policy $\pi(a|s)$ from all successful trajectories. By contrast, our method (GLiDE +BC) first segments the demonstrations and then learns mode-specific policies. Additionally, instead of mode-based imitation, we could also do planning to stay in the mode boundaries recovered by the classifier, since the system is fully-actuated with a single-integrator dynamics. Specifically, (GLiDE +Planning) uses RRT to compute waypoints to guide motion in non-convex mode 1 and then uses potential fields in convex polygons to generate trajectories that stay in the mode until entering the next mode. Table I shows that our methods perform slightly better than BC across different environments. However, when external perturbations are introduced, BC suffers the biggest performance degradation as recovery at the motion level without attention to mode boundaries may incur invalid transitions leading to task failures. The fact that (GLiDE +Planning) can almost maintain the same success rate despite perturbations validates the learned grounding.

Interpretability In the 2D environment, visualization of learned mode families can expose mode constraints and explain why some but not all perturbed demonstration replays fail the task execution.

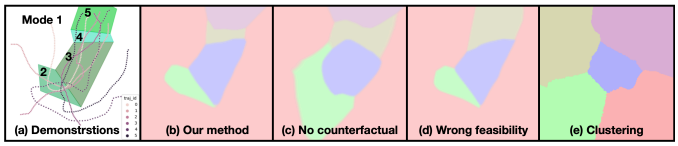

Fig. 5: Grounding of 2D navigation task. (a) Given six demonstrations that start in mode 1 and end in mode 5, visualized on top of the ground truth, (b) our method GLiDE recovers the underlying mode abstractions. (c) Without counterfactual data, GLiDE fails to learn precise boundaries. (d) Without a correct feasibility matrix (e.g. 4-mode instead of 5-mode), GLiDE results will miss modes. (e) Lastly, clustering the 2D state space to the nearest mode centers, discovered in the demonstrations by kmeans++, produces an incorrect modal structure.

Method	3-Mode (+perturb)	4-Mode (+perturb)	5-Mode (+perturb)
Behavior Cloning (BC)	0.967 (0.908)	0.814 (0.614)	0.810 (0.596)
GLiDE +BC	0.963 (0.887)	0.892 (0.753)	0.893 (0.753)
GLiDE +Planning	**0.996 (0.996)**	**0.987 (0.966)**	**0.991 (0.974)**

TABLE I: 2D navigation success rates for mode-agnostic imitation (BC) and mode-based (GLiDE) imitation or planning for environments consisting of 3, 4, and 5 modes. We test both perturbed and non-perturbed settings 1000 times and report the average success rate. For neural network-based BC policies, we report the average performance across 10 different random seeds.

B. Robosuite

Setup We test GLiDE across three tasks from Robosuite: placing a can in a bin (*can*), lifting a block (*lift*), and inserting a square nut into a peg (*square*). We use the default action and observation space of each environment unless an LLM suggests different features (e.g., relative distance to an object or keypoints). Since the manipulation tasks define underactuated systems, we use Eq. 3 for training.

Results: Mode classification To evaluate the mode classification accuracy, we manually define the ground truth modes for each environment (details in Appendix C). Table II shows

Method	Can	Lift	Square
GLiDE	**0.83**	**0.83**	**0.67**
GLiDE - Dynamics Loss	0.67	0.75	0.46
GLiDE - Prediction Loss	0.67	0.68	0.56
GLiDE - Feature Selection	0.55	0.70	0.57
Traj. Seg. Baseline	0.66	0.56	0.54

TABLE II: Ablation study of the influence of different loss terms and baselines on the mode classification accuracy based on overlap (percentage) with the ground truth.

Method	Can	Lift	Square
BC	0.93	0.99	0.38
BC (p)	0.20	0.18	0.03
GLiDE +BC	0.85	0.99	0.25
GLiDE +BC (p)	0.40	0.39	0.15

TABLE III: The success rate of mode-agnostic imitation (BC) drops more than that of mode-conditioned imitation (GLiDE +BC) after introducing perturbations (denoted by *p*).

the percentage of overlap between mode predictions from different methods and the ground truth mode segmentation. The results show that including all of the loss terms in our method achieves the best boundary alignment with the ground truth as visualized in Appendix C. Ablating the dynamics loss or the task prediction loss ($\mathcal{L}_{succ}$ and $\mathcal{L}_{fail}$) degrades the prediction accuracy as the classifier misses the precise location of important events such as dropping a grasped object. Comparing GLiDE to training without feature selection shows the importance of using an LLM to down-sample the feature space for efficient learning. A trajectory segmentation baseline using kmeans++ clustering on the features also underperforms GLiDE, highlighting the limitation of similarity-based segmentation methods.

Results: Task execution To show the learned grounding can help recover from perturbations, we compare a mode-agnostic BC baseline, which is trained on unsegmented successful demonstrations, to a mode-conditioned method (GLiDE +BC) described in Section II-D, where each per-model BC policy is augmented with a pseudo-attractor. While our method is insufficient to recover from all potential failures, our goal is to demonstrate how even a basic control strategy that leverages the underlying mode families can benefit policy learning in robotics. Table III summarizes the methods' performance without and with perturbations, which will randomly displace the end-effector or open the gripper. We see that for both methods, adding perturbations introduces some amount of performance drop. We find that the performance degradation for the BC baseline is much higher than with GLiDE +BC.

Interpretability In manipulation environments, it is challenging to directly visualize the mode families given the high-dimensional state space. However, exposing the mode families allows us to easily identify mode transition failures which can be used to generate post-hoc explanations of failures (e.g., videos on our website show invalid mode transitions associated with a task failure).

C. Real Robot Experiments: 2D Navigation and Scooping Tasks

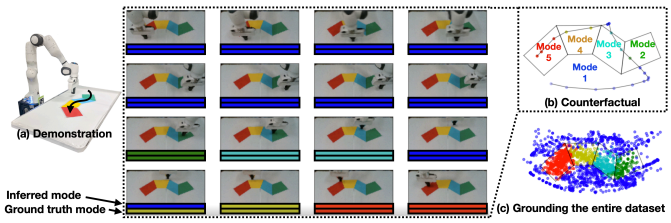

Fig. 6: Illustration of the real robot 2D navigation task (a), where the end-effector traces through a sequence of colored polygons. (b) shows a perturbed trajectory, overlaid on ground truth mode boundaries, experiences an invalid transition from mode 1 to mode 4. A vision-based classifier can predict from only pixels the inferred modes (first color bar) that match ground truth (second color bar) with high probability. (c) visualizes the mode prediction of individual image states seen in the dataset. The location of the scattered dots indicates where the images are recorded while the colors show the predictions, which are well-aligned with mode boundaries.

2D navigation To illustrate GLiDE can also learn grounding classifiers directly from vision inputs, we implement the simulated 2D navigation task on a real Franka robot, where the end-effector traces through a sequence of colored polygons in a plane. First, we record 20 human demonstrations through kinesthetic teaching in the end-effector's state space that start in various parts of mode 1 and end in mode 5 as seen in 6a. Second, we add end-effector perturbations to the demonstration replays to generate coverage over the entire tabletop area and record the perturbed trajectories in image sequences as seen in 6b. For this planar task, we use a simple reset mechanism that brings the end-effector back to one of the demonstrations' starting states after each rollout to collect data continuously. Since we can check if the end-effect is within the convex hull of any colored regions whose vertices are known, we log the mode sequence of each perturbed trajectory and automatically label if the trajectory is a successful task execution by checking if all mode transitions are feasible. Consequently, we were able to collect 2000 labeled trajectories in 2 hours continuously without human supervision. To learn a vision-based classifier, we switch from using multi-layer perceptrons (MLP) for state-space inputs to convolutional neural networks (CNN) to encode image inputs. Figure 6c shows the learned classifier can group image observations into correct modes according to the ground truth mode boundaries.

Marble scooping task The second task requires a spoon-holding robot to scoop marbles from a bowl and then transport at least one marble to a second bowl across the table. A typical robot implementation might require engineering a marble detector to check if the spoon is holding marbles and plan actions accordingly [36]. Instead, we learn a marble classifier on a wrist camera view to leverage LLM-based replanning as shown in Fig. 7. To collect successful executions, we record the end-effector's pose and wrist camera view as a human demonstrates scooping from various starting states. Since it is non-trivial to engineer a reset mechanism for this task, we ask humans to demonstrate various failures through kinesthetic teaching as well. To improve learning efficiency, we preprocess the raw wrist image to extract a mask corresponding to marble objects, where an empty spoon returns an empty mask. Specifically, we prompt an LLM for relevant object types to track, with which we employ the Segment Anything Model (SAM) [19] to generate segmentation masks. The classifier is then trained on a state representation consisting of end-effector poses and marble masks (details in Appendix C). To show the utility of the learned mode abstractions, we show a mode-agnostic policy cannot recover from task-level perturbations (dropping marbles) while a mode-based policy can leverage LLM to replan to ensure successful execution in Fig. 7. Videos can be found on our website https://yanweiw.github.io/glide/.

Learning abstractions from demonstrations. A large body of work focuses on learning action abstractions from language and interaction. This includes the grounding of natural language [7, 3, 2, 17, 30, 24], programs [31], and linear temporal logic (LTL) formulas [4, 34, 33]. In contrast to learning policies for individual action terms, this paper focuses on learning mode families in robot manipulation domains. These learned mode families enable us to construct robust policies under perturbation [36]. Furthermore, our framework is capable of

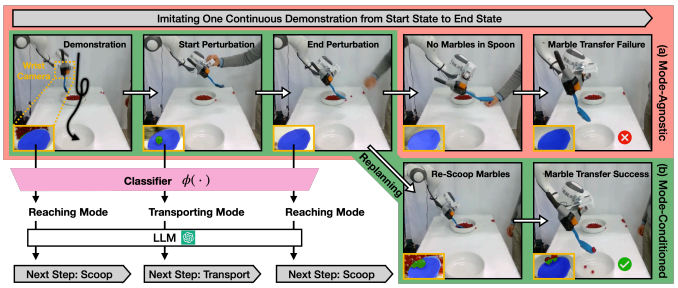

Fig. 7: Illustration of the robot scooping task. Here the perturbations are human-initiated (e.g., moving the spoon to drop or fail to grasp marbles). A mode-agnostic BC that imitates continuous motion demonstrations cannot replan at the task level when all marbles are dropped during the transporting mode (a). In contrast, our mode-conditioned policy enabled by the grounding classifier can leverage LLMs to replan given external perturbations (b).

recovering the underlying modes from a small number of *unsegmented* demonstrations. Through the use of synthetic noise [9, 37], we relieve humans from the burden of providing negative demonstrations that fail a task by automatically generating positive and negative variations of task executions.

Grounding language in robot behavior. With the rise of LLMs that can decompose high-level commands into sequences of actions, there has been much recent interest in the ability to ground these commands in embodied agents. Given that data from environment interactions (including human demonstrations) do not explicitly identify constraints and success criteria of a task, previous work has investigated how to infer affordances directly from observations [1]. Compared to prior work (e.g., [26]), our method does not require dense labels to learn a grounding operator. We are also not directly using large language models for planning [15, 21, 16]. Rather, we are using LLM to guide the discovery of mode abstractions in demonstrations, and as a result, we can also acquire a grounding operator for high-level language commands. In contrast to the discovery of language-conditioned skills [25, 12], which can consist of multiple modes, our mode decomposition occurs at a lower level and can explain why certain trajectories fail a task execution.

Counterfactuals. Counterfactuals describe hypothetical situations of alternative outcomes compared to the original data [5]. In other words, they are fake (non-human generated) data with a different result (e.g., failing a task instead of succeeding) [18]. In this paper, we define counterfactual perturbations as non-human-generated synthetic probes that test which parts of the time-series trajectory data [9] demonstrated by humans have implicit constraints, the violation of which will change the outcome of the successful human demonstrations.

Figure 8 visualizes data augmentation of generating additional successful trajectories (middle) and failing counterfactuals (right) from a few successful demonstrations (left). Figure 9 visualizes learned grounding for additional randomly generated 2D navigation environments with a 3-, 4- and 5-mode task structure.

We use the following heuristic rules to define the ground truth mode families to evaluate the grounding learned by GLiDE:

- *Can (3 modes):* the ground truth modes are reaching for the can (until the end effector makes contact with the can), transporting the can to the bin, and finally hovering about the target bin.
- *Lift (3 modes):* the ground truth modes are reaching for the cube (until the end effector makes contact with the cube), lifting the cube off the table, and finally moving to a certain height above the table.
- *Square (4 modes):* the ground truth modes are reaching for the nut (until the end effector makes contact with the nut), transporting the nut to the peg, aligning the nut above the peg, and finally lowering the nut into the assembled position.

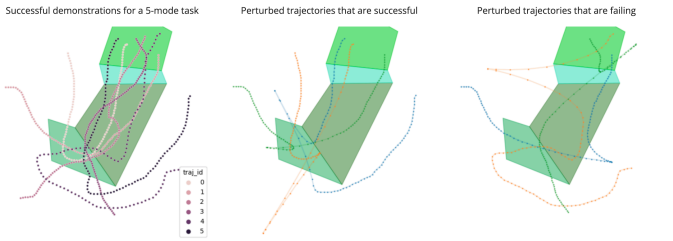

Fig. 8: Additional 2D grounding examples. Column (a) shows the ground truth mode segmentation and successful demonstrations. Column (b) shows the learned grounding and its percentage overlap with the ground truth. Columns (c-e) visualize the grounding learned without counterfactual data, a correct feasibility matrix, and task prediction loss respectively.

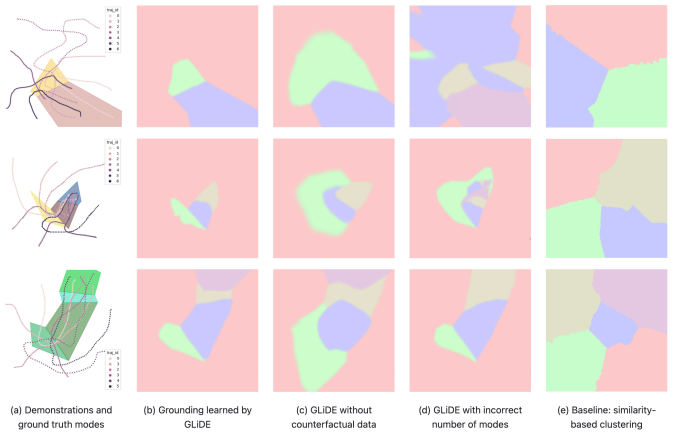

Fig. 9: Additional 2D grounding examples. Column (a) shows the ground truth mode segmentation and successful demonstrations. Column (b) shows the learned grounding and its percentage overlap with the ground truth. Columns (c-e) visualize the grounding learned without counterfactual data, a correct feasibility matrix, and task prediction loss respectively.

We assess the predicted and ground truth mode (based on the heuristics) for each sample in the robosuite demonstrations to compute accuracy (i.e., the percentage of samples where the predicted and ground-truth modes are the same). Fig. 10 shows the visualization of the mode segmentation from GLiDE, compared to the ground truth on the can placing task. Our model faithfully identifies all the modes and yields a high consistency with the modes defined by human-crafted rules.

We prompt LLM to generate a subset of features relevant to predicting task success: X, and Y locations of the robot end-effector in the robot base frame as well as the wrist camera

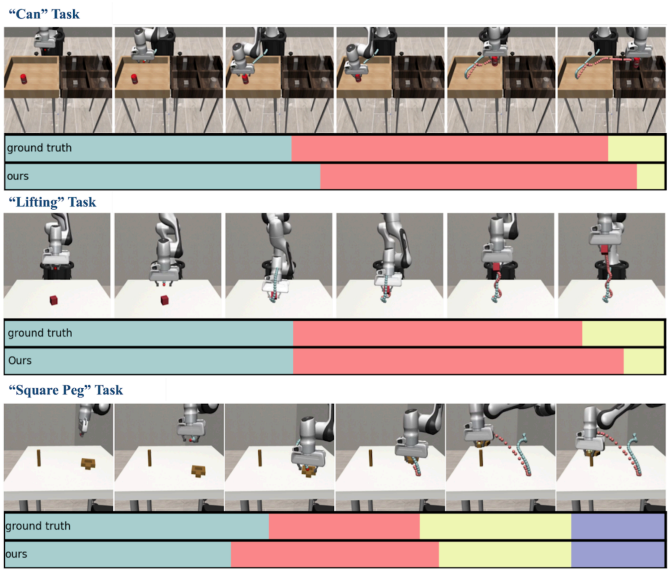

Fig. 10: Comparison on the robosuite can task between our method's segmented modes and the ground truth modes. Generally, our system is able to accurately recover mode boundaries.

mask. Due to a lack of contact sensors, we omit the scooping mode, and prompt the LLM to generate a plan: Reaching $\rightarrow$ Transporting $\rightarrow$ Dropping (assuming scooping is always successful when transitioning from the reaching mode to the transporting mode). The corresponding feasibility matrix is F^3. In Fig. 11 (top), we plot demonstrations in X and Y and use the color of the scattered plot to indicate ground truth modes (reaching is red, transporting is green, and dropping is blue). Examples of logged spoon masks along these trajectories are shown at the top. At the bottom, we visualize the learned classifier, which has correctly learned three modes (indicated by three distinct colors) by partitioning the space according to X and Y locations and the masks. Note that the location of the learned blue mode matches the dropping bowl location.

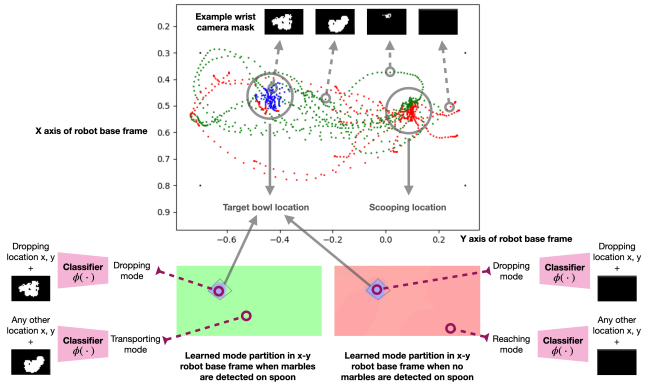

Fig. 11: Visualizing the state representation of the scooping task demonstrations and the learned mode partitions.