Points2Plans: From Point Clouds to Long-Horizon Plans with Composable Relational Dynamics

Yixuan Huang[1,2], Christopher Agia[1], Jimmy Wu[3], Tucker Hermans[2,4], Jeannette Bohg[1]
[1]Stanford University, [2]University of Utah, [3]Princeton University, [4]NVIDIA Research

Abstract: We present Points2Plans, a framework for composable planning with a relational dynamics model that enables robots to solve long-horizon manipulation tasks from partial-view point clouds. Given a language instruction and a point cloud of the scene, our framework initiates a hierarchical planning procedure, whereby a language model generates a high-level plan and a sampling-based planner produces constraint-satisfying continuous parameters for manipulation primitives sequenced according to the high-level plan. Key to our approach is the use of a relational dynamics model as a unifying interface between the continuous and symbolic representations of states and actions, thus facilitating language-driven planning from high-dimensional perceptual input such as point clouds. Whereas previous relational dynamics models require training on datasets of multi-step manipulation scenarios that align with the intended test scenarios, Points2Plans uses only single-step simulated training data while generalizing zero-shot to a variable number of steps during real-world evaluations. We evaluate our approach on tasks involving geometric reasoning, multi-object interactions, and occluded object reasoning in both simulated and real-world settings. Results demonstrate that Points2Plans offers strong generalization to unseen long-horizon tasks in the real world, where it solves over 85% of evaluated tasks while the next best baseline solves only 50%. Qualitative demonstrations of our approach operating on a mobile manipulator platform are made available at sites.google.com/stanford.edu/points2plans.

Keywords: Long-horizon planning, Robot manipulation, Semantic manipulation

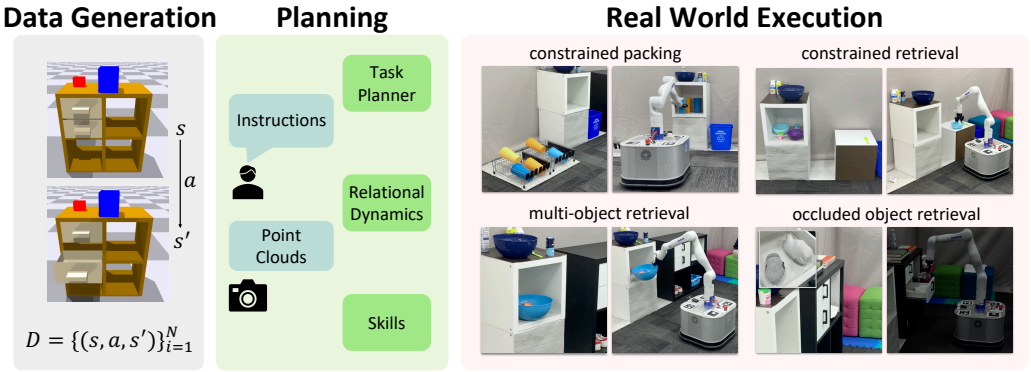

Figure 1: **Training, planning, and execution phases of Points2Plans. Left:** In simulation, we sample an environment state and execute a manipulation primitive at random to generate a dataset of single-step environment transitions and train a relational dynamics model. **Middle:** At planning time, Points2Plans receives a language instruction and a partial-view, segmented point cloud of the scene and then performs long-horizon planning in a hierarchical fashion with a task planner (e.g., a language model) and the learned relational dynamics model. If planning is successful, Points2Plans returns a sequence of manipulation primitives (what to execute) and their associated continuous parameters (how to execute them) for the given task. **Right:** Points2Plans executes its plan to solve a variety of unseen long-horizon tasks in the real world.

1 Introduction

Before robots can make their entry as general-purpose helpers in e.g., household environments, they must learn to solve *sequential manipulation* tasks in the presence of partial occlusions while receiving high-dimensional sensor data as input. Consider the "constrained packing" task shown in Fig. 1, where the robot must place all cups into the shelf without collision. To succeed, the robot has to reason about the long-horizon effects of its actions (e.g. what happens if the first cup is placed at the front of the shelf?) without perfect knowledge of object geometries or poses.

The most common paradigm for solving sequential manipulation tasks decomposes a task into a sequence of skills for the robot to execute [1, 2]. The open problem remains: how to sequence skills without being *myopic*; returning to our example, placing the first cup at the front of the shelf prevents future placements. Traditionally, this problem is addressed by task and motion planning (TAMP) systems, which perform a search for feasible solutions at the symbolic and geometric level [3, 4]. However, TAMP typically assumes access to explicit 3D object models and symbolic operators with predefined effects [5]; assumptions that may not hold in increasingly unstructured and partially observable environments. Other approaches leverage policy hierarchies to learn long-horizon strategies with reinforcement learning (RL) [6, 7, 8]. However, these approaches aim to learn skill sequencing strategies for each new long-horizon task, while we seek to compose skills through planning to solve a large set of downstream tasks [9, 10, 11]. We thereby ask: *How can we enable composable planning in high-dimensional observation spaces without predefined symbolic operators?*

In this paper, we argue that transformer-based relational dynamics (RD) [12, 13] is key to enabling composable, long-horizon planning directly from partial-view point clouds. RD models implicitly capture the symbolic and geometric effects of robot actions in a shared (object-centric) latent space, which facilitates goal-directed planning. We first propose an RD model architecture that requires only randomized single-step environment transitions (s,a,s') for training, but can be iteratively applied to predict long-horizon trajectories $(s_1,a_1,...,s_H)$ at plan time. Each single-step transition (s,a,s') corresponds to the states before and after executing a manipulation primitive (e.g., picking an apple from a basket and placing it on the table), and thus represents an abstraction over low-level trajectories executed on the robot [14]. Second, we introduce a sampling-based planning algorithm that selects robot actions that maximize the likelihood of symbolic goals predicted by the RD model. This algorithm uses a new rollout strategy that interweaves *delta-state* prediction of objects in the latent space with object pose updates in geometric space, resulting in greater accuracy over long-horizons compared to predicting absolute object states as in prior work [13]. Finally, we leverage large language models (LLMs) [15] to accelerate our planner by predicting candidate plan skeletons; in effect, significantly reducing the number of discrete skill sequences our planner must search through for any given task. The combination of these components forms the full Points2Plans planning framework.

Our contributions are three-fold: 1) A **relational dynamics model** that excels at long-horizon prediction of point cloud states without the need to train on multi-step data; 2) A **latent-geometric space dynamics rollout strategy** that significantly increases the horizons over which predicted point cloud states are reliable for planning; 3) A **planning framework**, Points2Plans, that integrates our RD model, rollout strategy, sampling-based planner, and task planner to solve complex long-horizon tasks. In extensive experiments, we demonstrate that Points2Plans generalizes to sequential manipulation tasks involving partial occlusions, long-horizon geometric dependencies, and multi-object interactions in both simulated and real-world settings.

2 Related Work

A standard approach to solving **long-horizon manipulation tasks** sequences manipulation *skills* [16, 17, 18, 19, 20, 21] according to a high-level plan produced by symbolic planners [22, 23, 24, 2, 25, 26], language models [1, 27, 28, 11, 29, 30], or combinations [31, 32, 33, 34, 35]. Reusability of the underlying skills should, in theory, support generalization to a variety of tasks. However, existing methods are often limited by myopic planning strategies or constraints resulting from skill design. For example, works employing visuomotor skills seldom consider the feasibility of skill sequences and hence evaluate tasks that do not require geometric reasoning [1, 2]. Conversely, works that optimize skill sequences for geometrically complex tasks

often rely on hand-crafted state representations [9, 11, 10]. Our work jointly addresses these limitations by enabling long-horizon lookahead planning from high-dimensional observations (i.e., 3D point clouds).

Alternative approaches **leverage policy hierarchies** to solve long-horizon manipulation tasks through options [36, 37, 38], parameterized action Markov decision processes [39, 40, 7, 41, 42], model-based RL [6, 43, 44], and meta-learning [45, 46]. Skill chaining has also been used to coordinate dependencies among skills in a sequence [47, 48]. These approaches attain strong performance within the distribution of tasks they are trained on, but may struggle to generalize to unseen tasks [9, 6]. Instead of training on long-horizon demonstration data, our approach relies on random single-step environment transitions to train a dynamics model, which is then used to compose skills for entirely new long-horizon tasks.

A number of works **learn dynamics models** for planning in high-dimensional observation spaces. Latent space dynamics are used for model-based control [49, 50, 51, 52, 53, 54], but predict state changes at comparatively small timescales. Graph neural networks can predict deformable [55, 56] and multi-object [57, 58, 59, 60, 61, 62] dynamics. Other works generate demonstration data via differentiable simulation to learn skill abstractions for deformable object planning from high-dimensional input [63, 64]. Several works learn RD models [12, 13, 65] that operate on 3D point clouds, but they lack *composability* (i.e., to support multi-step planning, they must be trained on multi-step trajectories) and are only demonstrated on tasks requiring up to three consecutive skills. Our method extends the task horizon over which RD predictions are reliable and enables the efficient sequencing of unseen skill sequences at test time.

Task and motion planning solves long-horizon tasks through symbolic and geometric reasoning [3, 4, 5, 66]. Perception modules can be used to alleviate TAMP's assumption on full state observability [67]. Other works learn vision-based planning heuristics [68, 69] or behavior policies [70, 71] from pre-computed TAMP solutions. We highlight two distinctions of our approach compared to TAMP: our system a) predicts symbolic effects instead of using predefined symbolic operators in e.g., PDDL [72] and b) plans in a latent space directly encoded from 3D partial-view point clouds. Our approach is related to learning symbolic operators [73, 74, 75, 76, 77] and object dynamics [9, 78] for long-horizon planning, but differs in the use of our RD model, which captures both symbolic and geometric effects of actions in a shared latent space.

3 Problem Setup

We aim to solve sequential manipulation tasks given segmented, partial-view 3D point clouds of the scene $\mathbf{o}_1$ and a natural language instruction l describing the task. Satisfying the instruction l entails achieving a goal configuration of objects (i.e., a goal state) $\mathcal{G}$ that can be expressed with predicates from a closed set $\mathcal{R}$. Predicates are boolean-valued functions that describe object properties e.g., $\texttt{Movable}(a)$,$\texttt{Openable}(a) \in \mathcal{R}$ and relationships among objects e.g., $\texttt{Above}(a,b)$,$\texttt{Inside}(a,b) \in \mathcal{R}$, including those that dictate action feasibility e.g., $\texttt{Blocking}(a,b) \in \mathcal{R}$. A ground predicate is a predicate expressed over specific object instances (e.g., $\texttt{Above}(cup,table)$), and a *fact* is an assertion of truth over a ground predicate (e.g., $\texttt{Above}(cup,table) = \text{true}$). Thus, we define the goal state as a conjunction of desired facts $\mathcal{G} = g_1 \wedge ... \wedge g_M$, where each fact g_j specifies a desired spatial relationship among objects. In this work, we assume that the closed set of predicates $\mathcal{R}$ is pre-specified, while noting that methods exist to learn predicates from data [77, 79].

Manipulation Primitives. To solve long-horizon tasks, we assume access to a library of manipulation primitives $\mathcal{L}^\phi = \{\phi^1, ..., \phi^K\}$. Each primitive ϕ^k takes as input continuous parameters $a^k \in \mathcal{A}^k$ and executes a trajectory on the robot [18]. For example, to pick up an object, the parameters a^{pick} could correspond to a target grasp pose in the object relative frame; instantiating the primitive $\phi^{\text{pick}}\left(a^{\text{pick}}\right)$ would move the robot's end-effector to the target pose and close its gripper. An *action* ψ^k is defined as a pair of a primitive and a parameter $\langle \phi^k, a^k \rangle$.

Perception. We assume access to two perception modules: a) a segmentation method that can return segmented point clouds $\mathbf{o}$; b) an object detector that returns the semantic class of each object. In this work, we use open-source models for segmentation [80] and detection [81].

The Planning Objective. Given an instruction l and segmented partial-view point clouds $\mathbf{o}_1$, our objective is to compute a plan $\tau = [\psi_1, ..., \psi_H]$ (we use range subscripts to denote sequences e.g., $\psi_{1:H}$) that when

executed maximizes the probability of the goal implied by instruction l:

$$\underset{\mathcal{G},\psi_{1:H}}{\operatorname{argmax}}\ p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1). \tag{1}$$

The first term defines the probability of observing an instruction l given observation $\mathbf{o}_1$ and the user's hidden logical goal $\mathcal{G}$. The second term defines the probability of achieving the logical goal $\mathcal{G}$ given the initial observation $\mathbf{o}_1$ and robot actions $\psi_{1:H}$. We refer to Appx. A.1 for further details.

4 Proposed Approach: Points2Plans

The planning objective in Eq. 1 can be optimized via a hierarchical approach [82] that first generates a *task plan* in the form of a sequence of primitives $\phi_{1:H}$ and then evaluates its feasibility when planning the parameters $a_{1:H}$ of the primitive sequence. We formulate our hierarchical planner via two distributions in Eq. 2 (the derivations of which are provided in Appx. A.2)

$$\underset{\mathcal{G},\psi_{1:H}}{\operatorname{argmax}}\ p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \approx \underset{G,\psi_{1:H}}{\operatorname{argmax}}\ q_1(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1)q_2(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1). \tag{2}$$

The first distribution $q_1(\phi_{1:H},\mathcal{G}|l,\mathbf{o}_1)$ represents the task planner, which serves two roles: a) proposing candidate task plans $\phi_{1:H}$ that are symbolically correct *w.r.t.* the instruction l and initial observation $\mathbf{o}_1$, and b) converting the instruction l into its corresponding goal state $\mathcal{G}$ used to ensure completion of the task. In this work, we use LLMs [15] to predict candidate task plans $\phi_{1:H}$ and goals $\mathcal{G}$ from instructions l and textual scene descriptions, while noting that other symbolic [13] and data-driven [69] alternatives are possible.

Given a candidate task plan, we must determine whether it can be feasibly executed in the environment and achieve the desired goal. Therefore, upon sampling $\tilde{\phi}_{1:H}$ and $\tilde{G}$ from the task planner $q_1(\phi_{1:H},\mathcal{G}|l,\mathbf{o}_1)$, we sample parameters $\tilde{a}_{1:H}$ from the second distribution $q_2(a_{1:H}|\tilde{\phi}_{1:H},\tilde{\mathcal{G}},\mathbf{o}_1)$ to approximately solve the optimization problem in Eq. 2. This second distribution represents the probability that parameters $\tilde{a}_{1:H}$ satisfy the goal $\tilde{\mathcal{G}}$ given observation $\mathbf{o}_1$ and task plan $\tilde{\phi}_{1:H}$. To obtain parameters $\tilde{a}_{1:H}$, we propose a long-horizon planning procedure with a transformer-based RD model. The full planning procedure is visualized in Fig. 2.

In the following sections, we outline our RD model architecture (Sec. 4.1), a hybrid rollout strategy for predicting point cloud states (Sec. 4.2), and finally, we present our full planning approach (Sec. 4.3).

4.1 Composable Relational Dynamics

Modeling the effects of actions on the environment (i.e., the *dynamics*) is essential for long-horizon planning. Yet, obtaining dynamics models that are both accurate and applicable to a wide range of downstream tasks is challenging for several reasons: a) they are difficult to define or learn with e.g., imperfect state knowledge or multi-object interactions; b) models trained on one distribution of long-horizon sequences may not generalize well to others. To address these challenges, we propose several design considerations for transformer-based RD models [13] that yield significant improvements in prediction accuracy and allow the model to be chained to predict entirely new long-horizon sequences. Our RD model is comprised of three components: an encoder Enc, a transformer-based dynamics model T, and a decoder Dec, all of which are jointly trained on single-step environment transitions (see Appx. A.6). We describe the details of each component below.

Encoder. The encoder Enc takes as input segmented point clouds $\mathbf{o}_t = o_t^1,...,o_t^M$ at timestep t and produces a factored, object-centric latent state $\mathbf{z}_t = z_t^1,...,z_t^M$, where M is the number of objects in the scene (which may vary across tasks). It embeds each per object segment using PointConv [83] and appends a learned positional embedding in PyTorch [84] to the resultant per object latent, giving $\mathbf{z}_t = Enc(\mathbf{o}_t)$.

Dynamics. We propose a *delta-dynamics* model T that takes as input the current latent state $\mathbf{z}_t$ and action $\psi_t = \langle \phi_t, a_t \rangle$, and predicts the *delta state* in the latent space as $\delta\mathbf{z_t} = T(\mathbf{z}_t,\psi_t)$. We use a transformer as the delta-dynamics model T since its inductive bias can represent interactions among the multiple objects in $\mathbf{z}_t$ as a result of action ψ_t. Our hypothesis is that it is easier to learn the relative effect $\delta\mathbf{z_t}$ of an action ψ_t than it is to directly predict the resulting *absolute state* $\mathbf{z}_{t+1}$ (i.e., hereafter referred to *absolute dynamics* [12, 13]), since the relative effects of actions might be similar across many states $\mathbf{z}_t$. We show in Sec. 5 that the choice of delta dynamics translates to notable improvements in pose and predicate prediction accuracy.

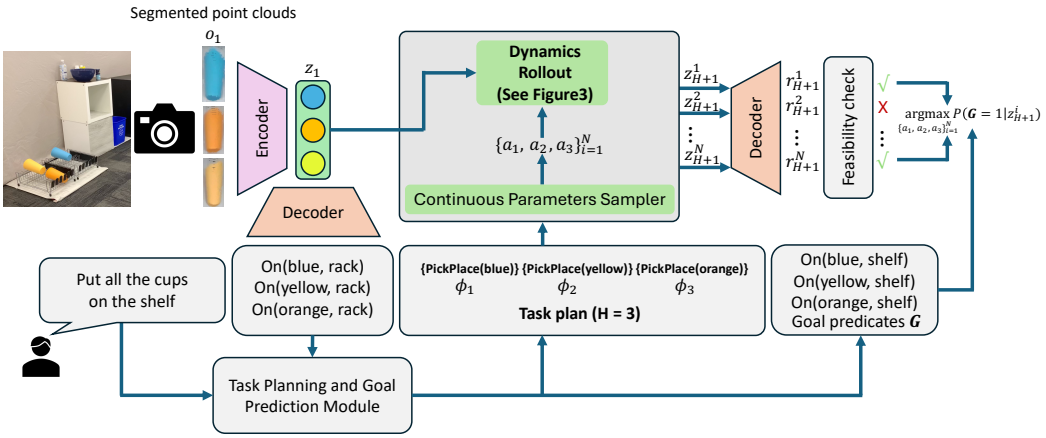

Figure 2: **Overview of Points2Plans.** A partial-view segmented point cloud $\mathbf{o}_1$ is first encoded into the (object-centric) latent state $\mathbf{z}_1$. The latent state $\mathbf{z}_1$ is then decoded into predicates that serve as environment context for the task planning and goal prediction module (e.g., an LLM), from which a task plan $\phi_{1:H}$ and a symbolic goal $\mathcal{G}$ are sampled. Points2Plans then invokes a sampling-based planning procedure to compute continuous parameters $a_{1:H}$ for the manipulation primitives in the task plan $\phi_{1:H}$. Infeasible plans (e.g., collisions) are rejected, and the plan that maximizes the goal likelihood in the final state $\mathbf{z}_{H+1}$ is returned.

Decoder. The decoder Dec consists of two heads: a relation decoder Dec_r and a pose decoder Dec_p. The relation decoder Dec_r predicts the probability of each ground predicate being true. More formally, if $\mathcal{U}$ represents the set of ground predicates, then $\mathbf{r}_t = Dec_r(\mathbf{z}_t) = \{p(u{=}\mathrm{true}\,|\,\mathbf{z}_t)\,|\,\forall u \in \mathcal{U}\}$. The probability $p(u{=}\mathrm{true}\,|\,\mathbf{z}_t)$ for one ground predicate $u \in \mathcal{U}$ is denoted by $Dec_r^u(\mathbf{z}_t)$. This decoder head can operate on any latent state $\mathbf{z}_t$; for instance, it can be used to detect facts at the initial state as $\mathbf{r}_1 = Dec_r(Enc(\mathbf{o}_1)) = Dec_r(\mathbf{z}_1)$. The pose decoder Dec_p takes as input a delta state in the latent space $\delta\mathbf{z}_t$ and predicts the relative pose change of all objects in the scene as $\delta\mathbf{p}_t = \delta p_t^1, \ldots, \delta p_t^M = Dec_p(\delta\mathbf{z}_t)$. Hence, this decoder can only be applied to delta states predicted by T.

4.2 Hybrid Rollout Strategy

We propose a hybrid latent-geometric space dynamics rollout strategy that uses the RD encoder Enc, dynamics T, and decoder Dec (described in Sec. 4.1) to predict the future states of a given plan $\tau = \psi_{1:H}$, i.e., based on the task plan $\phi_{1:H}$ and its continuous parameters $a_{1:H}$. The rollout strategy is visualized in Fig. 3.

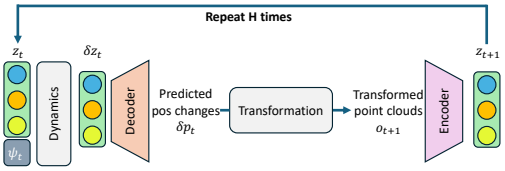

Figure 3: **Points2Plans hybrid rollout strategy.**

Let us consider the first timestep: the hybrid rollout strategy first encodes segmented point cloud $\mathbf{o}_1$ into the latent state $\mathbf{z}_1 = Enc(\mathbf{o}_1)$. Conditioned on the first action ψ_1, the delta state is then predicted as $\delta\mathbf{z}_1 = T(\mathbf{z}_1, \psi_1)$. The decoder Dec_p predicts the delta change in pose $\delta\mathbf{p}_1$. Finally, $\delta\mathbf{p}_1$ is used to transform the point clouds $\mathbf{o}_1$ to obtain $\mathbf{o}_2 = \omega(\delta\mathbf{p}_1)\mathbf{o}_1$. This process is repeated for all timesteps H in the plan resulting in the final point cloud $\mathbf{o}_{H+1}$ and latent $\mathbf{z}_{H+1}$ state. By interweaving latent and geometric state representations, our rollout strategy mitigates compounding prediction errors in the latent space.

4.3 Planning Action Sequences with Relational Dynamics

We outline our full approach to planning an action sequence $\psi_{1:H}$ from a language instruction l and the initial segmented point cloud of the scene $\mathbf{o}_1$ (visualized in Fig. 2). Our approach is hierarchical: we solve the optimization problem in Eq. 2 by first generating a candidate task plan $\tilde{\phi}_{1:H}$ with the LLM and then attempting to sample a set of continuous parameters $\tilde{a}_{1:H}$ that the robot can feasibly execute.

Given an instruction l with an initial observation $\mathbf{o}_1$, we sample $\tilde{\phi}_{1:H}$ and $\tilde{\mathcal{G}}$ from $q_1(\phi_{1:H}, \mathcal{G}\,|\,l, \mathbf{o}_1)$. In practice, we use a shooting-based method [11], which queries an LLM few-shot to predict N task plans

$\{\tilde{\phi}^i_{1:H_i}\}_{i=1}^N$ and their corresponding symbolic goals $\{\tilde{\mathcal{G}}^i\}_{i=1}^N$ (prompt details in Appx. A.5). For each task plan $\tilde{\phi}_{1:H}$ and goal $\tilde{\mathcal{G}}$ predicted by the LLM, we then seek to generate primitive parameters $\tilde{a}_{1:H}$ from distribution $q_2(a_{1:H}|\tilde{\phi}_{1:H},\tilde{\mathcal{G}},\mathbf{o}_1)$ that will approximately maximize the objective in Eq. 1. We formulate the planning process for parameters $\tilde{a}_{1:H}$ as the following constrained optimization problem:

$$\tilde{a}^*_{1:H} = \underset{\tilde{a}_{1:H}\sim q_2}{\operatorname{argmax}} \prod_{g\in\tilde{\mathcal{G}}} Dec^g_r(\mathbf{z}_{H+1}) \tag{3}$$

$$\texttt{subject to} \quad Dec^c_r(\mathbf{z}_t) < \epsilon, \forall c\in\mathcal{C}, \forall t\in 1,...,H+1 \tag{4}$$

$$\texttt{where} \quad \mathbf{z}_t = Enc(\mathbf{o}_t), \forall t\in 1,...,H+1 \tag{5}$$

$$\delta\mathbf{z}_t = T\Big(\mathbf{z}_t, \langle\tilde{\phi}_t,\tilde{a}_t\rangle\Big), \forall t\in 1,...,H \tag{6}$$

$$\delta\mathbf{p}_t = Dec_p(\delta\mathbf{z}_t), \forall t\in 1,...,H \tag{7}$$

$$\mathbf{o}_{t+1} = \omega(\delta\mathbf{p}_t)\mathbf{o}_t, \forall t\in 1,...,H \tag{8}$$

We optimize Eq. 3 to maximize the probability of achieving goal predicates $\tilde{\mathcal{G}}$ using sampling-based optimization techniques [85]. The relation decoder Dec_r is used to compute the probability of a ground goal predicate g holding true in the final latent state $\mathbf{z}_{H+1}$, which we denote with $Dec^g_r(\mathbf{z}_{H+1})$. $\mathcal{C}$ in Eq. 4 represents the set of all feasibility-related ground predicates, such as $\texttt{Blocking}(bowl,cup)$. During optimization, we reject parameter sequences $\tilde{a}_{1:H}$ that violate feasibility constraints, i.e., ground predicates $c\in\mathcal{C}$ whose probability (predicted by the relation decoder $Dec^c_r(\mathbf{z}_t)$) exceeds a calibrated threshold ϵ_c. For example, we would reject a plan that attempts to grasp a *cup* if $\texttt{Blocking}(bowl,cup)$ holds true. The remaining equations (Eq. 5-Eq. 8) correspond to the steps of our hybrid rollout strategy (Sec. 4.2).

For each candidate task plan $\tilde{\phi}_{1:H}$ and goal $\tilde{\mathcal{G}}$ predicted by the LLM, we compute their corresponding parameters $\tilde{a}^*_{1:H}$ via optimization (Eq. 3). If the success probability $\prod_{g\in\tilde{\mathcal{G}}} Dec^g_r(\mathbf{z}_{H+1})$ resulting from the optimal plan $\psi^*_{1:H} = \langle\tilde{\phi}_{1:H},\tilde{a}^*_{1:H}\rangle$ exceeds a success threshold ϵ_s (e.g., 90%), we execute the plan on the robot. However, if no task plan predicted by the LLM is successful or constraint-satisfying, we fall back to a graph search strategy that enumerates all possible primitive sequences up to a specified search depth (as in [13]). This ensures that more task plans will be tested should the LLM fail to produce a correct plan.

5 Experiments

We conduct experiments to test the following questions: **Q1:** Can Points2Plans generalize to unseen long-horizon tasks despite only being trained on single-step environment transitions? **Q2:** Does our hybrid rollout strategy and delta-dynamics model improve prediction accuracy compared to previous RD rollout formulations? **Q3:** Does Points2Plans outperform approaches that sequence skills without predicting dynamics or reasoning about feasibility? **Q4:** Can LLMs improve the planning efficiency of Points2Plans? We generate a dataset of over 36,000 random executions of manipulation primitives in IsaacGym [86] to train our RD model (see Appx. A.6 for training details) and use GPT-4 [87] as the LLM for all experiments.

Dynamics Planning Baselines. We test the performance of Points2Plans against five planning baselines. Points2Plans−Geo (read "minus geo") uses the same RD model as Points2Plans but performs rollouts exclusively in the latent space, i.e., without the point cloud transformation in our hybrid rollout strategy (Sec. 4.2). Conversely, Points2Plans−Delta uses our hybrid rollout strategy but employs an absolute-dynamics model to predict the absolute state $\mathbf{z_t}$ of objects instead of the delta state $\delta\mathbf{z_t}$. eRDTransformer [13] represents the current state-of-the-art RD planning approach, which uses a latent space rollout strategy and an absolute-dynamics model. We train eRDTransformer using single-step transitions instead of multi-step transitions for fair comparison. Pairwise-RD [49] is equivalent to eRDTransformer except it only captures pairwise object interactions with a multi-layer perceptron (MLP) instead of multi-object interactions with transformers. Greedy selects actions $a_{1:H}$ to avoid collisions without dynamics prediction and long-horizon planning.

Task Planning Baselines. We test the performance of Points2Plans under an LLM task planner and two baselines. Search performs an exhaustive graph search as in [13] for task planning. Points2Plans−Feasibility uses the LLM without access to the feasibility-related ground predicates, e.g., $\texttt{Blocking}(a,b)$.

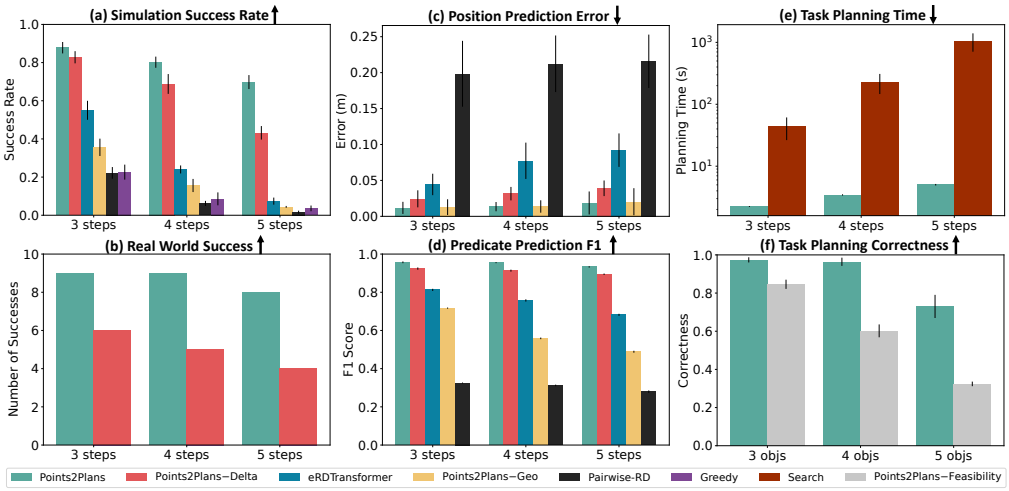

Figure 4: **Simulation and real-world results** for the **Constrained Packing** (a-d) and **Constrained Retrieval** (e-f) tasks. As task complexity increases, Points2Plans significantly outperforms baselines in terms of planning success rate (a-b), position prediction error (c), and predicate classification accuracy (d). Interfacing Points2Plans with an LLM task planner increases planning efficiency (e) and correctness (f). Planning time is shown on a logarithmic scale. Errors bars denote standard deviations across 500 trials.

Experimental Tasks. We evaluate our approach and baselines across a suite of sequential manipulation tasks (Fig. 1). **Constrained Packing** tasks the robot with shelving multiple objects in a spatially constrained environment (e.g., a kitchen cupboard). To succeed, the robot must carefully plan the placement positions of the objects so as to avoid collisions. We compare Points2Plans to the RD baselines on this task as it requires accurate RD predictions of geometric and symbolic states. **Constrained Retrieval** tasks the robot with retrieving target objects in a constrained environment. To succeed, the robot must identify and remove objects that occlude the target objects before retrieving them. We compare Points2Plans to the task planning baselines on this task as it requires the planner to infer the logically correct task plan based on the initial state. **Multi-object Retrieval** tasks the robot with retrieving an object inside a container (e.g., a bowl) in a constrained environment. Here, the robot must first remove the container from the constrained environment before grasping the object from inside the container. This task tests our planner's ability to reason about multi-object interactions and nested geometric dependencies. **Occluded Object Retrieval** tasks the robot with retrieving objects in a dark environment (i.e., without perception) given the history of states and actions up until the timestep t at which the lights are turned off. To succeed, the robot must plan from its *memory* of object positions and relations encoded in the latent state $\mathbf{z}_t$. We present quantitative results on the **Constrained Packing** and **Constrained Retrieval** tasks, and qualitative results on the **Multi-object Retrieval** and **Occluded Object Retrieval** tasks.

6 Results

Simulation Experiments. We compare Points2Plans against all planning baselines on the **Constrained Packing** task to evaluate the effect of the RD model and rollout strategy on planning performance. We run 500 trials for each combination of planning horizon and approach. To measure the planning performance, we report the success rate, position prediction error, and predicate prediction F1 score.

Results shown in Fig. 4a demonstrate that Points2Plans generalizes to unseen long-horizon tasks more effectively than the baselines (Q1). Comparing Points2Plans and Points2Plans–Geo in Fig. 4d, we observe that our hybrid rollout strategy contributes greatly to predicate prediction accuracy over long horizons (Q2). Moreover, comparing Points2Plans with Points2Plans–Delta, eRDTransformer, and Pairwise-RD in Fig. 4c highlights the importance of our delta-dynamics model, as the baselines (which employ an absolute-dynamics model) exhibit a larger accumulation of position prediction error over increasing prediction horizons (Q2). Finally, we observe that the Greedy approach performs significantly worse than Points2Plans in Fig. 4a, indicating that multi-step planning is required for the long-horizon tasks we consider (Q3).

Real-World Experiments. We evaluate Points2Plans against Points2Plans–Delta (the next best-performing baseline) in the real world. We run 10 trials of each method per task. The results in Fig. 4b show that Points2Plans solves over 85% of long-horizon tasks and significantly outperforms Points2Plans–Delta, which solves only 50% of the tasks. Fig. 5 (top row) illustrates how the baseline fails to plan collision-free placements for multiple objects, due to prediction errors from its RD model. In contrast, Fig. 1 and Fig. 5 show that Points2Plans effectively generalizes to various real-world tasks without fine-tuning (Q1). Demonstrations of approach on a mobile manipulator are available at sites.google.com/stanford.edu/points2plans.

Task Planning Ablation. We evaluate the performance of Points2Plans when configured with different task planning strategies in the **Constrained Retrieval** task. We run 500 trials of each approach per task. Fig. 4e shows that the planning time of Points2Plans increases only linearly with the LLM task planner, whereas the Search task planner (enumerating all possible discrete parameters) results in an exponential increase in planning time *w.r.t.* to the task horizon (Q4). In Fig. 4f, we see that the Points2Plans–Feasibility baseline struggles to predict feasible task plans, highlighting the importance of providing feasibility-related predicates to the LLM task planner as in Points2Plans. Plan executions of Points2Plans and Points2Plans–Feasibility are shown in the bottom row of Fig. 5. Here, the baseline fails to remove occluding objects before attempting to grasp the target object behind them, while Points2Plans infers a feasible task plan based on feasibility-related predicates detected by the RD model.

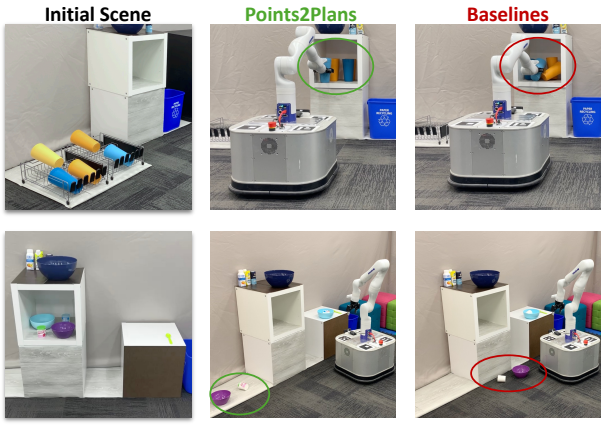

Figure 5: Points2Plans generalizes to unseen long-horizon tasks, whereas the baselines struggle to find collision-free plans.

7 Conclusion

In this work, we study the problem of solving sequential manipulation tasks from partial-view point clouds and language instructions. We present a long-horizon planning framework, Points2Plans, that uses transformer-based relational dynamics to sequence manipulation skills and coordinate their geometric dependencies. In experiments, we show that interleaving additive, delta-state predictions in the latent space with rigid-body transformations in the geometric space leads to more accurate predictions of point cloud states over long horizons. As a result, our relational dynamics model can accurately learn the effects of robot skills from a dataset of random, single-step transitions, and then compose the skill effects at planning time to solve multi-step tasks. We deploy Points2Plans on a mobile manipulator platform and demonstrate that it can generalize to diverse real-world tasks such as shelving kitchenware, retrieving occluded objects, and planning from memory.

Limitations. Points2Plans executes its plans in an open-loop fashion, i.e., without considering feedback from the environment. Exploring closed-loop strategies for refining plans or correcting execution failures via replanning are possible points of extension. Furthermore, our framework assumes a fixed set of predicates to learn skill effects. Drawing from predicate learning techniques [79, 88] could improve the generality of our framework, e.g., facilitating planning in an open-world setting. Finally, while we demonstrate faster planning with LLMs, their task planning performance degrades with longer tasks and more complex instructions [89]. Including more sophisticated LLM-based task planning strategies [11] would improve the overall robustness of our planner.

8 Acknowledgements

This work was supported by NSF Award #2024778, by DARPA under grant N66001-19-2-4035, and by a Sloan Research Fellowship. Toyota Research Institute and Toshiba provided funds to support this work.

References

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] B. Wu, R. Martin-Martin, and L. Fei-Fei. M-ember: Tackling long-horizon mobile manipulation via factorized domain transfer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11690–11697. IEEE, 2023.

[3] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, pages 1930–1936, 2015.

[4] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 440–448, 2020.

[5] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4: 265–293, 2021.

[6] 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 *2021 IEEE international conference on robotics and automation (ICRA)*, pages 6206–6213. IEEE, 2021.

[7] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:21847–21859, 2021.

[8] L. X. Shi, J. J. Lim, and Y. Lee. Skill-based model-based reinforcement learning. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2262–2272. PMLR, 14–18 Dec 2023.

[9] C. Agia, T. Migimatsu, J. Wu, and J. Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.

[10] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[11] 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.

[12] Y. Huang, A. Conkey, and T. Hermans. Planning for Multi-Object Manipulation with Graph Neural Network Relational Classifiers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2209.11943.

[13] 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 (T-RO)*, 2024. URL https://arxiv.org/pdf/2305.10857.pdf.

[14] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 701–714. PMLR, 14–18 Dec 2023.

[15] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[16] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[17] B. C. Da Silva, G. Konidaris, and A. G. Barto. Learning parameterized skills. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1443–1450, 2012.

[18] J. Felip, J. Laaksonen, A. Morales, and V. Kyrki. Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks. *Robotics and Autonomous Systems*, 61(3):283–296, 2013.

[19] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[20] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song. XSkill: Cross embodiment skill discovery. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=8L6pHd9aS6w.

[21] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton. Structdiffusion: Language-guided creation of physically-valid structures using unseen objects. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[22] L. P. Kaelbling and T. Lozano-Pérez. Learning composable models of parameterized skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 886–893. IEEE, 2017.

[23] D.-A. Huang, D. Xu, Y. Zhu, A. Garg, S. Savarese, L. Fei-Fei, and J. C. Niebles. Continuous relaxation of symbolic planner for one-shot imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2635–2642. IEEE, 2019.

[24] W. Yuan, C. Paxton, K. Desingh, and D. Fox. Sornet: Spatial object-centric representations for sequential manipulation. In *Conference on Robot Learning*, pages 148–157. PMLR, 2022.

[25] S. Cheng and D. Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 2023.

[26] N. Kumar, T. Silver, W. McClinton, L. Zhao, S. Proulx, T. Lozano-Pérez, L. P. Kaelbling, and J. Barry. Practice makes perfect: Planning to learn skill parameter policies. In *Robotics: Science and Systems (RSS)*, 2024.

[27] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=3R3Pz5i0tye.

[28] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[29] 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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[30] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[31] D. Xu, R. Martín-Martín, D.-A. Huang, Y. Zhu, S. Savarese, and L. F. Fei-Fei. Regression planning networks. *Advances in neural information processing systems*, 32, 2019.

[32] C. Wang, D. Xu, and L. Fei-Fei. Generalizable task planning through representation pretraining. *IEEE Robotics and Automation Letters*, 7(3):8299–8306, 2022.

[33] 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.

[34] 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 preprint arXiv:2304.11477*, 2023.

[35] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2024 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2024.

[36] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[37] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[38] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[39] W. Masson, P. Ranchod, and G. Konidaris. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[40] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Efficient bimanual manipulation using learned task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1149–1155. IEEE, 2020.

[41] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7477–7484. IEEE, 2022.

[42] K. Fang, P. Yin, A. Nair, H. R. Walke, G. Yan, and S. Levine. Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=esOrVR_8-rc.

[43] D. Shah, A. T. Toshev, S. Levine, and brian ichter. Value function spaces: Skill-centric state abstractions for long-horizon reasoning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vgqS1vkkCbE.

[44] L. X. Shi, J. J. Lim, and Y. Lee. Skill-based model-based reinforcement learning. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=iVxy2eO601U.

[45] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[46] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8565–8574, 2019.

[47] Y. Chen, C. Wang, L. Fei-Fei, and K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=2Qrd-Yw4YmF.

[48] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=K5-J-Espnaq.

[49] C. Paxton, C. Xie, T. Hermans, and D. Fox. Predicting stable configurations for semantic placement of novel objects. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=5DjX89Wyhk-.

[50] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[51] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[52] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[53] P. Sundaresan, J. Wu, and D. Sadigh. Learning sequential acquisition policies for robot-assisted feeding. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=o2wNSCTkq0.

[54] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022.

[55] H. Shi, H. Xu, S. Clarke, Y. Li, and J. Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=69y5fzvaAT.

[56] 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. *The International Journal of Robotics Research*, page 02783649231219020, 2023.

[57] M. Chang, T. Ullman, A. Torralba, and J. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Bkab5dqxe.

[58] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[59] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning*, pages 4470–4479. PMLR, 2018.

[60] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[61] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Conference on robot learning*, pages 1755–1768. PMLR, 2023.

[62] 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 *Conference on Robot Learning*, pages 1582–1601. PMLR, 2021.

[63] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *International Conference on Learning Representation (ICLR)*, 2022.

[64] X. Lin, C. Qi, Y. Zhang, Z. Huang, K. Fragkiadaki, Y. Li, C. Gan, and D. Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tyxyBj2w4vw.

[65] Y. Huang, J. Yuan, C. Kim, P. Pradhan, B. Chen, L. Fuxin, and T. Hermans. Out of Sight, Still in Mind: Reasoning and Planning about Unobserved Objects with Video Tracking Enabled Memory Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[66] B. Vu, T. Migimatsu, and J. Bohg. Coast: Constraints and streams for task and motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[67] 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 *2022 International Conference on Robotics and Automation (ICRA)*, pages 1940–1946. IEEE, 2022.

[68] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 9563–9569. IEEE, 2020.

[69] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Robotics: Science and Systems (RSS)*, 2020.

[70] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint. Learning geometric reasoning and control for long-horizon tasks from visual input. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 14298–14305. IEEE, 2021.

[71] M. Dalal, A. Mandlekar, C. R. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=QNPuJZyhFE.

[72] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[73] B. Ames, A. Thackston, and G. Konidaris. Learning symbolic representations for planning with parameterized skills. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 526–533. IEEE, 2018.

[74] 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.

[75] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3182–3189. IEEE, 2021.

[76] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7): 866–894, 2021.

[77] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12120–12129, 2023.

[78] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4166–4173. IEEE, 2022.

[79] 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.

[80] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[81] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lL3lnMbR4WU.

[82] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[83] W. Wu, Z. Qi, and L. Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019.

[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[85] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

[86] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=fgFBtYgJQX_.

[87] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[88] A. Ahmetoglu, B. Celik, E. Oztop, and E. Ugur. Discovering predictive relational object symbols with symbolic attentive layers. *IEEE Robotics and Automation Letters*, 2024.

[89] 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 preprint arXiv:2402.01817*, 2024.

[90] K. Rawlik, M. Toussaint, and S. Vijayakumar. On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In *Robotics: Science and Systems*, 2012.

[91] A. Conkey and T. Hermans. Planning under uncertainty to goal distributions. *Arxiv Preprint*, 2020. URL http://arxiv.org/abs/2011.04782.

[92] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[93] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems (RSS)*, 2024.

[94] J. Y. Gil and R. Kimmel. Efficient dilation, erosion, opening, and closing algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1606–1617, 2002.

A Appendix

Overview

The appendix provides additional details and results. First, we present detailed derivations for our planning objective (Appx. A.1) and sampling distributions (Appx. A.2). Second, we provide the details of the planning and optimization (Appx. A.3). Third, we include extra experimental details (Appx. A.6 to Appx. A.4). Finally, we provide information on implementation (Appx. A.9), failure cases (Appx. A.10), hardware (Appx. A.11), generalization experiments (Appx. A.12), and robustness to noisy segmentations (Appx. A.13). Qualitative results are available at sites.google.com/stanford.edu/points2plans.

A.1 Generative Model and Problem Formulation

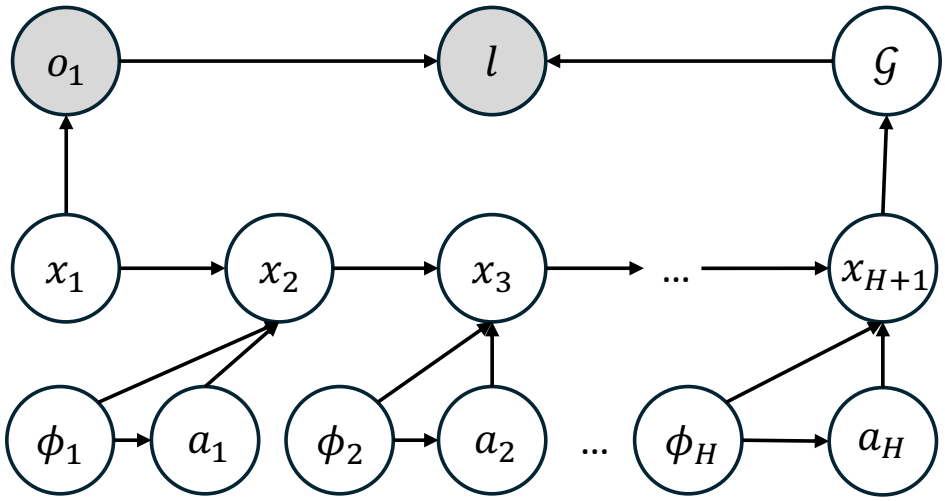

Figure 6: A causal Bayes net to derive Eq. 1. $\mathcal{G}$ represents the goal predicates, l is the language instruction, o_1 is the initial observation, $\phi_{1:H}$ are the task plans, $a_{1:H}$ are the continuous parameters, and $\mathbf{x}_{1:H}$ represent world states (including predicates $\mathbf{r}_{1:H}$ and positions $\mathbf{p}_{1:H}$. Shaded nodes represent observed variables.

Figure 6 shows a causal Bayes net defining the relevant variables to our planning problem. Recall that a Bayesian network defines a factorization via conditional independence over the joint probability distribution of all variables in the model. In our model, the joint distribution is thus, $p(\mathcal{G},\mathbf{o}_1,l,\psi_{1:H},\mathbf{x}_{1:H+1}) = p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{o}_1\,|\,\mathbf{x}_1)p(\mathbf{x}_1)\prod_{k=1}^{H}p(\mathbf{x}_{k+1}\,|\,\mathbf{x}_k,a_k,\phi_k)p(a_k\,|\,\phi_k)p(\phi_k)$.

Here $\mathbf{o}_1$ defines the observed world observation at time step 1. We define the observed language instruction as l and the unobserved goal predicates as $\mathcal{G}$. Our model assumes that the user has a goal in mind that the robot is capable of achieving. However, instead of explicitly writing this goal into a command prompt for the robot, the user provides a natural language command. As such the robot must infer the underlying goal predicates conditioned on the language instruction and the world observation shared between the user and robot.

The variable $\psi_{1:H}$ defines the robot's plan (sequence of primitives and continuous parameters), while $\mathbf{x}_k$ defines the world state at time k. We represent the state dynamics and action priors in the format common to that used in the planning as inference literature e.g. [90, 91].

While one could fully separate inference of the goal from the robot planning problem, by treating them as a joint problem we ensure that the robot only infers goals that are feasible for it to achieve. This matches our assumption that the human operator is non-adversarial and providing instructions the robot should be able to perform. The robot's planning task is thus to infer both the goal predicates, $\mathcal{G}$ and plan $\psi_{1:H}$ that achieves this goal. While we are primarily concerned with finding the plan, identifying the goal predicates provides a fixed target for our planner. This goal can also be helpful if we want to replan online to correct for execution errors. As an alternative, we could examine marginalizing out the goal variable by integrating over all possible goals. This would require us to use a form of planning to goal distributions [91], which we defer to future work.

Given this model associated with the causal graph in Fig. 6, we can now derive the planning objective of Eq. 1 through the following steps. (Note that Eq. 1-Eq. 8 appear in the main paper.)

$$\underset{\psi_{1:h},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G},\mathbf{o}_1,l,\psi_{1:H},\mathbf{x}_{1:H+1}) \tag{9}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{o}_1\,|\,\mathbf{x}_1)p(\mathbf{x}_1)\prod_{k=1}^{H} p(\mathbf{x}_{k+1}\,|\,\mathbf{x}_k,a_k,\phi_k)p(a_k\,|\,\phi_k)p(\phi_k) \tag{10}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{o}_1\,|\,\mathbf{x}_1)p(\mathbf{x}_1)p(\mathbf{x}_{H+1}\,|\,\mathbf{x}_1,\psi_{1:H})p(\psi_{1:H}) \tag{11}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{x}_1\,|\,\mathbf{o}_1)p(\mathbf{o}_1)p(\mathbf{x}_{H+1}\,|\,\mathbf{x}_1,\psi_{1:H})p(\psi_{1:H}) \tag{12}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{x}_{H+1}\,|\,\psi_{1:H},\mathbf{o}_1)p(\psi_{1:H})p(\mathbf{o}_1) \tag{13}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\mathbf{x}_{H+1})p(\mathbf{x}_{H+1}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{14}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{15}$$

The first step is simply applying the factorization of the Bayes net. Eq. 11 makes two changes to remove the product over time steps. First we apply the definition $p(\psi_{1:H}) = \prod_{k=1}^{H} p(a_k\,|\,\phi_k)p(\phi_k)$, which we name the plan prior. Second we integrate over (i.e. marginalize out) state trajectories as a function of the actions sequence (i.e. plan) to encode the distribution over terminal states $\mathbf{x}_{H+1}$ as a function of the initial state and plan. For Eq. 12 we then apply the definition of conditional distributions to the initial state prior and observation function. This allows us to then marginalize out the initial state variable $\mathbf{x}_1$ in Eq. 13. The next step removes the prior on the initial observation as it is constant and assumes the prior on plans is uniform and thus also constant. The final step then marginalizes out all possible terminal states to recover our problem as stated in Eq. 1.

A.2 Approximate Sampling Distributions

We now turn to the derivation of our approximate sampling distributions $q_1\,(\phi_{1:H},\mathcal{G},|\,l,\mathbf{o}_1)$ and $q_2\,(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)$. We start by taking the first term in Eq. 1 and use Bayes' theorem (Eq. 17) to condition on the observed language instruction l

$$\underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1)p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{16}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\mathcal{G}\,|\,l,\mathbf{o}_1)p(l\,|\,\mathbf{o}_1)}{p(\mathcal{G}\,|\,\mathbf{o}_1)} p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{17}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\mathcal{G}\,|\,\mathbf{o}_1)} p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{18}$$

where we can simplify the numerator in Eq. 17 to Eq. 18 since the value of l is known and thus $p(l\,|\,\mathbf{o}_1)$ is constant. We now turn our attention to the second term in Eq. 1 and Eq. 18. Here we again use Bayes' theorem and the definitions of conditional probability distributions to rearrange terms from Eq. 19 through

Eq. 21:

$$\underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\mathcal{G}\,|\,\mathbf{o}_1)} p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{19}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\mathcal{G}\,|\,\mathbf{o}_1)} \frac{p(\psi_{1:H}\,|\,\mathcal{G},\mathbf{o}_1) p(\mathcal{G}\,|\,\mathbf{o}_1)}{p(\psi_{1:H}\,|\,\mathbf{o}_1)} \tag{20}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G}\,|\,l,\mathbf{o}_1) \frac{p(\psi_{1:H}\,|\,\mathcal{G},\mathbf{o}_1)}{p(\psi_{1:H}\,|\,\mathbf{o}_1)} \tag{21}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G}\,|\,l,\mathbf{o}_1) \frac{p(a_{1:H},\phi_{1:H}\,|\,\mathcal{G},\mathbf{o}_1)}{p(a_{1:H},\phi_{1:H}\,|\,\mathbf{o}_1)} \tag{22}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G}\,|\,l,\mathbf{o}_1) \frac{p(\phi_{1:H}\,|\,\mathcal{G},\mathbf{o}_1)}{p(\phi_{1:H}\,|\,\mathbf{o}_1)} \frac{p(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)}{p(a_{1:H}\,|\,\phi_{1:H},\mathbf{o}_1)} \tag{23}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G}\,|\,l,\mathbf{o}_1) \frac{p(\phi_{1:H}\,|\,\mathcal{G},l,\mathbf{o}_1)}{p(\phi_{1:H}\,|\,\mathbf{o}_1)} \frac{p(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)}{p(a_{1:H}\,|\,\phi_{1:H},\mathbf{o}_1)} \tag{24}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(\mathcal{G}\,|\,l,\mathbf{o}_1) \frac{p(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\mathcal{G}\,|\,l,\mathbf{o}_1) p(\phi_{1:H}\,|\,\mathbf{o}_1)} \frac{p(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)}{p(a_{1:H}\,|\,\phi_{1:H},\mathbf{o}_1)} \tag{25}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\phi_{1:H}\,|\,\mathbf{o}_1)} \frac{p(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)}{p(a_{1:H}\,|\,\phi_{1:H},\mathbf{o}_1)} \tag{26}$$

where Eq. 22 comes from applying the definition of the action components and Eq. 23 comes from the fact that skills $\phi_{1:H}$ can be chosen before selecting their parameters $a_{1:H}$. The equality in Eq. 24 results from the existing conditional independence relations allowing us to introduce l without changing the values of the probability distribution. Eq. 25 results from the definition of conditional distributions. Finally, this allows us to cancel the term appearing in both the numerator and denominator resulting in Eq. 26. These two terms are then the distributions we approximate with our sampling distributions $q_1(\cdot)$ and $q_2(\cdot)$.

Hence we can now summarize the derivation above as the following relationships

$$\psi^*_{1:H}, G^* = \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} p(l\,|\,\mathcal{G},\mathbf{o}_1) p(\mathcal{G}\,|\,\psi_{1:H},\mathbf{o}_1) \tag{27}$$

$$= \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} \frac{p(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1)}{p(\phi_{1:H}\,|\,\mathbf{o}_1)} \frac{p(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1)}{p(a_{1:H}\,|\,\phi_{1:H},\mathbf{o}_1)} \tag{28}$$

$$\approx \underset{\psi_{1:H},\mathcal{G}}{\operatorname{argmax}} q_1(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1) q_2(a_{1:H}\,|\,\phi_{1:H},\mathcal{G},\mathbf{o}_1) \tag{29}$$

where the approximation in Eq. 29 is made by assuming a uniform distribution over actions given the initial observation (o.e. the denominator becomes constant). We can then approximately solve the planning objective in Eq. 1 by sequentially generating samples from the two approximate sampling distributions as $\tilde{\phi}_{1:H}, \tilde{G} \sim q_1(\phi_{1:H},\mathcal{G}\,|\,l,\mathbf{o}_1)$ followed by $\tilde{a}_{1:H} \sim q_2\Big(a_{1:H}\,\Big|\,\tilde{\phi}_{1:H},\tilde{G},\mathbf{o}_1\Big)$.

A.3 Planning and Optimization Details

We use a shooting-based planner to determine the actions $\psi_{1:H}$ from an initial observation $\mathbf{o}_1$ and a language instruction l. We provide the details of the shooting-based Points2Plans planner in Alg. 1.

Note that if no plan predicted by the LLM is successful or constraint-satisfying, we fall back to a search-based strategy that enumerates all possible primitive sequences up to a specified search depth (as in [13]), optimizes them with Eq. 3, and checks if the plan satisfies any goal predicted by the LLM. (Note that Eq. 1-Eq. 8 appear in the main paper.) In practice, we find that the planner seldom falls back to the search-based strategy; however, it ensures that more primitive sequences will be tested should the LLM fail to produce a correct plan.

Furthermore, we provide more details about the constrained optimization (Eq. 3). The optimization process includes encoding the point clouds into the latent states (Eq. 5), the *delta-state* predictions by the *delta-dynamics* (Eq. 6), decoding the *delta-state* in latent space to relative pose changes with the pose decoder Dec_p (Eq. 7) , and point cloud transformations (Eq. 8).

Algorithm 1 Shooting-based Points2Plans planner

globals: LLM, Enc, Dec_r^g, Dec_r^c, T, ω, $q_2(a_{1:H} \mid \phi_{1:H}, \mathcal{G}, \mathbf{o}_1)$
function SHOOTING(l, $\mathbf{o}_1$, C)
  $\{\tilde{\phi}^i_{1:H_i}\}_{i=1}^N, \{\tilde{\mathcal{G}}^i\}_{i=1}^N \sim \text{LLM}(l, \mathbf{o}_1)$ ▷ Generate task plans and goals
  for $i = 1...N$ **do**
    $C = \{\}$ ▷ Initiate candidate set for each task plan and goal
    $\{\tilde{a}^j_{1:H_i}\}_{j=1}^K \sim q_2\left(\tilde{a}_{1:H_i} \,\middle|\, \tilde{\phi}^i_{1:H_i}, \tilde{\mathcal{G}}^i, \mathbf{o}_1\right)$ ▷ Sample actions
    for $j = 1...K$ **do**
      $\mathbf{z}_1 = Enc(\mathbf{o}_1)$ ▷ Encode initial observation
      $\mathbf{z}_1^j = \mathbf{z}_1$
      $\mathbf{o}_1^j = \mathbf{o}_1$
      for $t = 1...H_i$ **do**
        $\delta \mathbf{z}_t^j = T\left(\mathbf{z}_t^j, \langle \tilde{\phi}_t, \tilde{a}_t^j \rangle\right)$ ▷ *Delta-dynamics* function
        $\delta \mathbf{p}_t^j = Dec_p\left(\delta \mathbf{z}_t^j\right)$
        $\mathbf{o}_{t+1}^j = \omega(\delta \mathbf{p}_t^j)\mathbf{o}_t^j$ ▷ Point clouds transformations
        $\mathbf{z}_{t+1}^j = Enc\left(\mathbf{o}_{t+1}^j\right)$ ▷ Encode transformed point clouds
        if $Dec_r^c\left(\mathbf{z}_{t+1}^j\right) >= \epsilon$ **then**
          raise `collision found, break` ▷ Collision found, reject this sequence $\tilde{a}^j_{1:H_i}$
        end if
        if $t == H$ **then** ▷ No collision found
          $C \leftarrow C \cup \{j\}$ ▷ Add this sequence to candidate set
        end if
      end for
    end for
    if $C != \emptyset$ **then**
      $j^* = \text{argmax}_{j \in C} \prod_{g \in \tilde{\mathcal{G}}_i} Dec_r^g(\mathbf{z}^j_{H_i+1})$
      return $\tilde{\phi}^i_{1:H_i} \tilde{a}^{j^*}_{1:H_i}$ ▷ Return the task plan with the continuous parameters
    end if
  end for
  raise `LLM failure, fall back to Search`
end function

A.4 Predicates Definition

Our system includes unary predicates and binary predicates.

For unary predicates, our system encodes whether a segment is movable (e.g., a shelf in a cupboard is not movable while an object on the shelf is movable), whether a segment is a drawer, and whether the drawer is opened or closed.

For binary predicates, our system includes two kinds. First, our system includes nine spatial predicates: *left, right, front, behind, above, below, contact, boundary,* and *inside*. The definitions of these predicates are the same as [65]. Second, we define feasibility-related predicates to indicate the feasibility of each object. We define two feasibility-related predicates: *blocking-behind* and *blocking-inside*. We define *blocking-behind*(a, b) as true if behind(b, a), below(a, high-surface), below(b, high-surface), above(a, low-surface), and above(b, low-surface), meaning both a and b are in a constrained environment, and b is behind a. We define *blocking-inside*(a, b) as true if inside(b, a), below(a, high-surface), below(b, high-surface), above(a, low-surface), above(b, low-surface), meaning both a and b are in a constrained environment and b is inside a.

A.5 LLM Prompt Details

Our prompts include prompt templates (black), LLM output (orange), and in-context examples (grey). Placeholders, denoted by braces, are substituted with task-related objects for different scenarios.

The in-context examples are toy examples of the tasks that the LLM solves at test time. These toy examples describe the usage semantics of the available primitives and predicates, and help constrain the LLM output.

I am the reasoning system of a mobile manipulator robot operating in a household environment. Given 1) an instruction from a human user and 2) the current symbolic state of the environment, I will predict a set of possible symbolic goals that the robot could achieve to fulfill the user's instruction.

Definitions:

- Symbolic states and symbolic goals are defined as a set of predicates expressed over specific objects.

- The term 'predicate' refers to an object state (e.g., Opened(cabinet)) or a relationship among objects (e.g., On(cup, shelf)).

The robot can perceive the following information about the environment: - The objects in the environment - The states of individual objects - The relationships among objects

The robot can detect the following states of individual objects: - Opened(a): Object a is opened - Closed(a): Object a is closed

The robot can detect the following relationships among objects: - On(a, b): Object a is on object b - Inside(a, b): Object a is in object b

There may be multiple symbolic goals that fulfill the user's instruction. Therefore, I will format my output in the following form:

Goals: List[List[str]]

Rules: - I will output a set of symbolic goals as a list of lists after 'Goals:'. Each nested list represents one goal - I will not output all possible symbolic goals, but the most likely goals that fulfill the user's instruction - If there are multiple symbolic goals that fulfill the instruction, I will output the simplest goals first"

I am the task planning system of a mobile manipulator robot operating in a household environment. Given 1) an instruction from a human user, 2) the current symbolic state of the environment, and 3) a set of possible symbolic goals that the robot could achieve to fulfill the user's instruction, I will predict a set of task plans that the robot should execute to satisfy the symbolic goals.

Definitions:

- Symbolic states and symbolic goals are defined as a set of predicates expressed over specific objects.

- The term 'predicate' refers to an object state (e.g., Opened(cabinet)) or a relationship among objects (e.g., On(cup, shelf)).

- A task plan is a sequence of actions that the robot can execute (e.g., Pick(cup, table), Place(cup, shelf))

The robot can perceive the following information about the environment:

- The objects in the environment

- The states of individual objects

- The relationships among objects

The robot can detect the following states of individual objects:

- Opened(a): Object a is opened - Closed(a): Object a is closed

The robot can detect the following relationships among objects:

- On(a, b): Object a is on object b - Inside(a, b): Object a is in object b

The robot can execute the following actions:

- Pick(a, b): The robot picks object a from object b

- Place(a, b): The robot places object a on or in object b

- Open(a): The robot opens object a

- Close(a): The robot closes object a

Action preconditions:

- If the robot is already holding an object, it CANNOT Pick, Open, or Close another object

- The robot CAN ONLY Place an object that it is already holding

There may be multiple symbolic goals that fulfill the user's instruction. Therefore, I will format my output in the following form:

Plans: List[List[str]]

Rules:

- I will output a set of task plans as a list of lists after 'Plans:'. Each nested list represents one task plan

- I will output one task plan for each symbolic goal. Hence, each goal and its corresponding plan will be located at the same index in the 'Goals' and 'Plans' lists

- I will only output task plans that are feasible with respect to the defined action preconditions.

```
Instructions: Put all the objects on the shelf

Objects: ['{object 1}', '{object 2}', '{object 3}', 'ground', 'shelf']

Predicates: ['On({object 1}, ground', 'On({object 2}, ground', 'On({object 3},
ground']

Goals: ['On({object 1}, shelf', 'On({object 2}, shelf', 'On({object 3}, shelf']

Plans: ['Pick({object 1}, ground)', 'Place({object 1}, shelf)', 'Pick({object 2},
ground)', 'Place({object 2}, shelf)', 'Pick({object 3}, ground)', 'Place({object 3},
shelf)']
```

```
Instructions: Retrieve object 1.

Objects: ['{object 1}', '{object 2}', '{object 3}', '{object 4}', 'ground', 'shelf']

Predicates: ['On({object 1}, shelf', 'On({object 2}, shelf', 'On({object 3}, shelf',
'On({object 4}, shelf', 'Blocking({object 3}, {object 4})', 'Blocking({object 3},
{object 1})', 'Blocking({object 2}, {object 1})', 'Blocking({object 4}, {object 1})']

Goals: [ '{On(object 2}, ground', '{On(object 3}, ground', '{On(object 4}, ground',
'{On(object 1}, ground']

Plans: ['Pick({object 3}, shelf)', 'Place({object 3}, ground)', 'Pick({object 2},
shelf)', 'Place({object 2}, ground)', 'Pick({object 4}, shelf)', 'Place({object 4},
ground)', 'Pick({object 1}, shelf)', 'Place({object 1}, ground)']
```

In-context Examples:

```
Instructions: Put object 1 on the sink.

Objects: ['object 1', 'sink', 'kitchen table']

Predicates: ['On(object 1, kitchen table']

Goals: ['On(object 1, sink']

Plans: ['Pick(object 1, kitchen table)', 'Place(object 1, sink)']
```

```
Instructions: Get me object 1 from the drawer. I'm in the bedroom. Don't leave the
drawer open.

Objects: ['object 1', 'object 2', 'drawer', 'closet', 'bed']

Predicates: ['Inside(object 1, drawer)', 'Inside(object 2, closet)', 'Closed(drawer)',
'Closed(closet)']

Goals: ['On(object 1, bed)', 'Closed(drawer)']

Plans: ['Open(drawer)', 'Pick(object 1, drawer)', 'Place(object 1, bed)',
'Close(drawer)']
```

```
Instructions:  Bring me object 2.  I'm sitting on the reading chair by the coffee ta-
ble.

Objects:  ['object 1', 'object 2', 'bookshelf', 'reading chair', 'coffee table']

Predicates:  ['On(object 1, object 2', 'On(object 2, bookshelf']

Goals:  [['On(object 2, coffee table)'], ['On(object 2, reading chair)']]

Plans:  [ ['Pick(object 1, object 2)', 'Place(object 1, bookshelf)', 'Pick(object
2, bookshelf)', 'Place(object 2, coffee table)'], ['Pick(object 1, object 2)',
'Place(object 1, bookshelf)', 'Pick(object 2, bookshelf)', 'Place(object 2, reading
chair)'] ]
```

```
Instructions:  Please retrieve object 1.

Objects:  ['object 1', 'object 2', 'shelf', 'ground']

Predicates:  ['On(object 1, shelf', 'On(object 2, shelf']

Goals:  [['On(object 1, shelf)'], ['On(object 2, shelf) ']]

Plans:  [ ['Pick(object 1, shelf)', 'Place(object 1, ground)'], ['Pick(object 2,
shelf)', 'Place(object 2, ground)', 'Pick(object 1, shelf)', 'Place(object 1, ground)',
] ]
```

A.5.1 Connections between RD Models and LLMs

We have several connections as the interface between RD models and LLMs. First, given the predicates *above*(A, B) and *contact*(A, B), then *on*(A, B) holds true, and vice versa. Second, given the plans as pick(A, D) and place(A, C) from LLMs, the RD models will receive this plan as pick-and-place(A,C).

A.6 Dataset Generation and Training Details

We generate the training datasets in the IsaacGym [86] simulator. First, we generate a variable number of randomized objects (size and pose) and save the object pose, segmented point cloud, and predicate as $(\mathbf{o}_t,\hat{\mathbf{r}}_t,\hat{\mathbf{p}}_t)$. Then we randomly execute a primitive in the simulator and save the primitive ψ_t. We teleport the objects to model the effects of each primitive. After the primitive execution, we record the post-action scene as $(\mathbf{o}_{t+1},\hat{\mathbf{r}}_{t+1},\hat{\mathbf{p}}_{t+1})$. The dataset contains more than 36,000 primitive executions. We show several single-step simulation executions in Fig. 7.

We set up one camera in simulation to generate the segmented point clouds. Due to the generalization ability of our RD framework to different view points, we can position the real-world camera at different view angles, as long as the object point clouds are within a suitable range to ensure decent quality. For example, we use the Realsense D435 camera for real-world experiments, with an ideal range of 0.3m to 3m. Note that we focus on the critical segments of environmental point clouds (e.g., horizon surfaces for the cupboard and drawers for the table), as not all segments are visible due to the partial-view nature of the input point clouds.

We define loss functions with four terms for each transition $(\mathbf{o}_t,\hat{\mathbf{r}}_t,\hat{\mathbf{p}}_t,\psi_t,\mathbf{o}_{t+1},\hat{\mathbf{r}}_{t+1},\hat{\mathbf{p}}_{t+1})$ in the training datasets. First, we obtain $\mathbf{z}_t = Enc(\mathbf{o}_t)$ and $\mathbf{z}_{t+1} = Enc(\mathbf{o}_{t+1})$. To enable our framework to detect the current predicates, we define the cross-entropy loss between the currently detected predicates and the ground-truth predicates: $L_{cp} = CE(Dec_r(\mathbf{z}_t),\hat{\mathbf{r}}_t) + CE(Dec_r(\mathbf{z}_{t+1}),\hat{\mathbf{r}}_{t+1})$. Second, to enable the model to accurately predict the change of pose, we define the second loss term as $L_{pos} = a\cdot\sqrt{b\cdot||\delta\mathbf{p}_t - (\hat{\mathbf{p}}_{t+1} - \hat{\mathbf{p}}_t)||}$. We use two parameters, a, b, to balance L_{pos} with other loss terms like L_{pd}. In practice, we use a = 5 and b = 12. Third, to regularize the latent states, we first obtain the predicted latent states as $\mathbf{z}'_{t+1} = \mathbf{z}_t + \delta\mathbf{z}_t$. We define the regularization loss term as $L_{reg} = ||\mathbf{z}_{t+1} - \mathbf{z}'_{t+1}||_2^2$. Fourth, to predict the

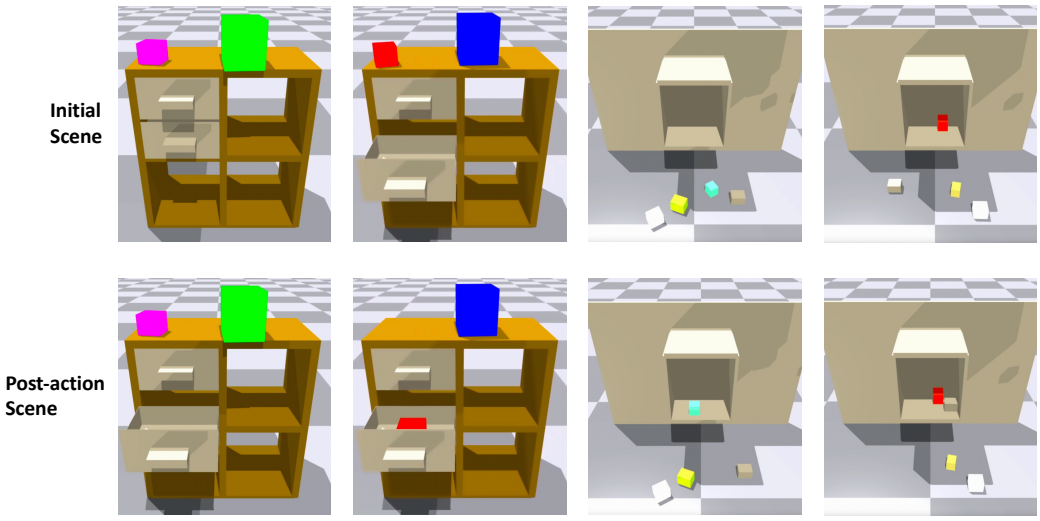

Figure 7: We show several examples of single-step primitive executions in simulation. The first two columns show examples of drawers, while the next two columns show examples of the constrained cupboard with different numbers of objects.

future predicates, we define a cross-entropy loss between the predicted predicates and the ground-truth predicates as $L_{fp} = CE(Dec_r^q(\mathbf{z}'_{t+1}), \hat{\mathbf{r}}_{t+1})$. We train our framework end-to-end with the sum of these four loss terms as $L = L_{cp} + L_{pos} + L_{reg} + L_{fp}$ using the Adam optimizer with a learning rate of $1e{-}4$. We only train our framework with single-step transitions while it can solve long-horizon planning problems in a composable way.

A.7 Baseline Comparison Details

We show the details of how baselines fail in the "constrained packing" task and the "constrained retrieval" task in Fig. 8. Please refer to the supplemental video for the demos.

A.8 Primitives Definition

We use three primitives in this paper: pick-and-place, pick-and-toss, and open/close. Since we have a mobile manipulator, we separate the movements of the mobile base and the Kinova arm. Based on the objects to manipulate, we first move the mobile base to a reachable space for the arm to manipulate the objects. Then, we run the arm planner to manipulate the objects.

Pick-and-place is defined as first grasping the object and then placing it on the supporting surface. For the grasp, we use the point cloud center of each segment to generate the grasps. We grasp the center of the objects except for large objects. For large objects, we use a heuristics offset. For example, we grasp the side of a bowl instead of the center.

For the placement, we generate a placement height based on the surface height plus a height offset. For the toss, we first move the base to a position at a fixed distance from the target, then execute the predefined toss trajectory.

For the open/close actions, we first determine the handle center using the segmented point clouds. Then we move the robot to the pre-open/close position. After this, the arm executes the motion with continuous parameters encoding how much the drawer will open or close.

A.9 Neural Network Implementation Details

Our RD model is composed of three components: an encoder Enc, a transformer-based dynamics model T, and a decoder Dec. We describe the details of each component below.

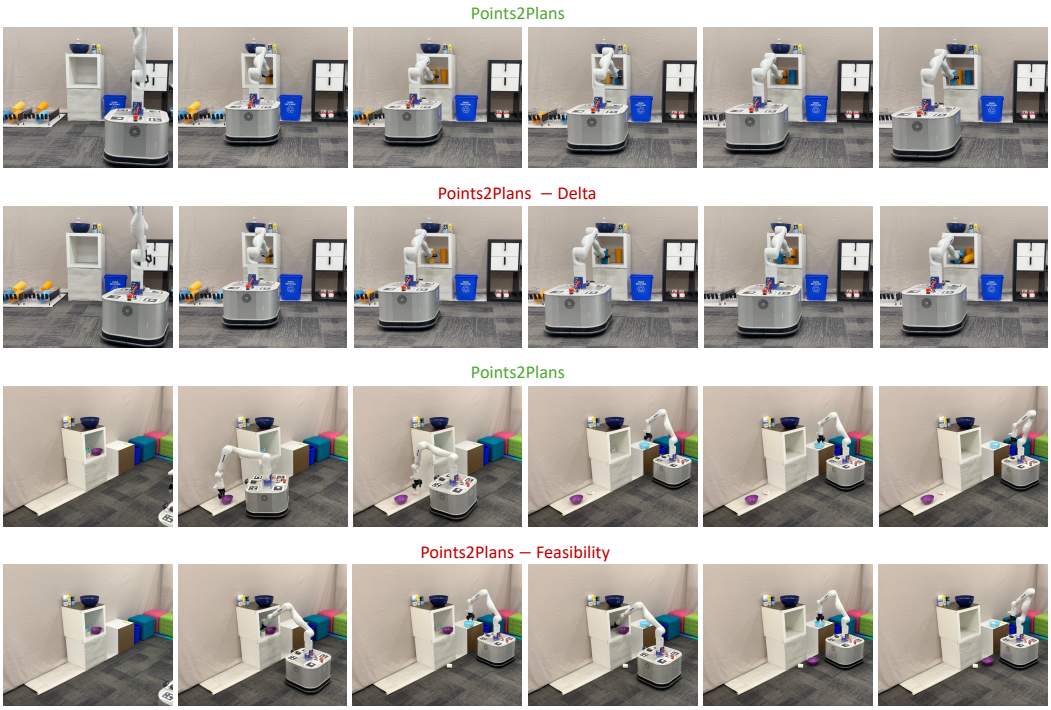

Figure 8: We show two failure cases of baselines. The first two rows demonstrate that Points2Plans succeeds while Points2Plans–Delta fails in the "constrained packing" task. The next two rows show that Points2Plans succeeds while Points2Plans–Feasibility fails in the "constrained retrieval" task.

Encoder: We first use the farthest point sampling method to downsample each point cloud to 128 points. Based on the input as segmented point clouds $\mathbf{o}_t = o_t^1, ..., o_t^M$ at timestep t, we first use a PointConv [83] to get per object features as $P_t^i = PointConv(o_t^i)$. The PointConv model we use incorporates three set abstraction layers. Each abstraction layer receives input points data and input points position data. The output from each layer consists of sampled points position data and sampled points feature data, with the input and output points position data having 3 channels. The first set abstraction layer has 128 points with 8 samples, utilizing a bandwidth of 0.1. It employs an MLP with 3+3 input channels, 32 output channels, and a kernel size of 1. The second layer has 64 points with 16 samples and a bandwidth of 0.2, using an MLP with 32+3 input channels, 64 output channels, and a kernel size of 1. The third layer is a group_all layer, generating 128-dimension features per segment with a bandwidth of 0.4, and utilizes an MLP with 64+3 input channels, 128 output channels, and a kernel size of 1.

Then we concatenate per object feature with the positional embedding of each object in PyTorch [84] as $z_t^i = P_t^i \oplus I_i$ where $I_i = Emb_{pos}(i)$. Each positional embedding has 128-dimension features. Next, we combine the per object latent into an object-centric latent state $\mathbf{z}_t = z_t^1, ..., z_t^M$, where each z_t^i has 256 features.

Dynamics: The *delta-dynamics* model T takes the input as $\mathbf{z}_t$ and $\psi_t = \langle \phi_t, a_t \rangle$. ϕ_t includes skill id si, manipulated obj id mi, and placement surface id pi. For each si, we use a different dynamics model T_{si} with a transformer. For the transformers, we utilize 2 sub-encoder layers, 2 heads in the multi-head attention models, and an input/output model size of 256.

We encode each a_t with an action encoder (MLP_{si}) as $a_t^m = I_{mi} \oplus MLP_{si}(a_t)$, where I_{mi} encodes which object this primitive will operate on. We further use the placement id to represent which surface to place the object on, as $a_t^p = I_{pi} \oplus MLP_{si}(a_t)$. If there is no surface to place, for example, in an open drawer action, we use zero embeddings for I_{pi}. The action encoder is a two-layer MLP, with each layer containing 128 neurons. Then the dynamics model T_{si} takes the input as M+2 tokens $z_t^1, ..., z_t^M, a_t^m, a_t^p$. We discard the action tokens at the output head and obtain the output $\delta \mathbf{z_t} = \delta z_t^1, ..., \delta z_t^M$

Decoder: Based on the latent state $\mathbf{z}_t$, we use different MLPs for different output heads. First, we use one MLP for unary predicate prediction: $\mathbf{r}_t^u = Dec_r^u(\mathbf{z}_t)$. Second, we use one MLP for constrained binary predicates prediction: $\mathbf{r}_t^c = Dec_r^c(\mathbf{z}_t)$. Third, we use one MLP for spatial binary predicate prediction: $\mathbf{r}_t^s = Dec_r^s(\mathbf{z}_t)$.

For Dec_r^u, we use a two-layer MLP with a hidden layer of 64 neurons. The output contains 3 bits. The first bit encodes whether the segment is a shelf or an object. The second and the third bits encode whether the segment is a drawer and whether the drawer is open, respectively. For Dec_r^c, we use a three-layer MLP and each hidden layer contains 64 neurons. The output contains 2 bits representing *blocking-behind* and *blocking-inside*. For Dec_r^s, we use a three-layer MLP with each hidden layer containing 64 neurons. The output contains 9 bits representing 9 different spatial predicates.

The pose decoder Dec_p takes as input a *delta-state* in the latent space $\delta\mathbf{z}_t$ and predicts the relative pose change of all objects in the scene as $\delta\mathbf{p}_t = \delta p_t^1, ..., \delta p_t^M = Dec_p(\delta\mathbf{z}_t)$. Hence, this decoder can only be applied on *delta-state* predicted by T.

For Dec_p, we use a two-layer MLP with one hidden layer containing 64 neurons. The output head contains 2 bits representing $\delta x, \delta y$. For z, we encode this parameter as part of our discrete parameter for the supporting surface, as shown in the primitive definitions in Sec. A.8. LLMs will generate it as part of the task plan.

For the MLPs, we use Sigmoid in the output head of predicate decoders Dec_r^u,Dec_r^c, and Dec_r^s since these decoders output binary variables. For all other MLPs, we use ReLU as the activation function.

A.10 Failure Cases Analysis

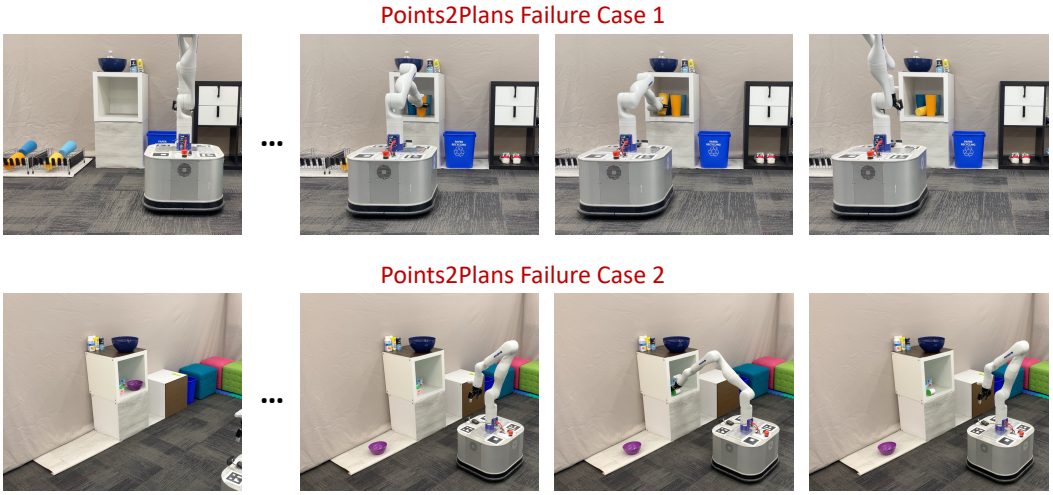

Figure 9: Two failure cases of Points2Plans. The first row shows a failure case due to an unstable placement in the "constrained packing" task. The second row demonstrates that Points2Plans fails in the "constrained retrieval" task because of a failed grasp.

We show two failure cases of Points2Plans in Fig. 9. These failure cases are caused by primitive execution failures. They highlight the limitations of open-loop execution in Points2Plans and motivate the incorporation of closed-loop policies [92, 93] in future work. Please refer to our website (sites.google.com/stanford.edu/points2plans) for detailed executions of these two failure cases.

A.11 Hardware Setup

The models are trained on a standard workstation with a single GPU (NVIDIA GeForce RTX 3090 Ti, 24 GB). All real-world experiments are conducted on a mobile platform with a custom mobile base and a Kinova arm. We use a RealSense D435 camera for perception in the real world.

A.12 Generalization to Unseen Scenarios

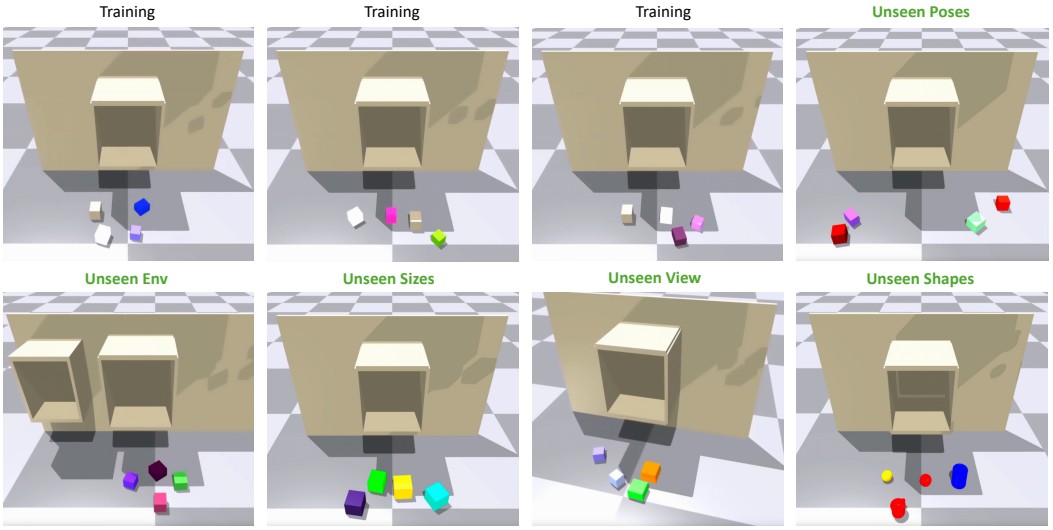

Figure 10: Examples of our training dataset and some test dataset with unseen poses, environments, sizes, view, and shapes.

Table 1: Generalization to Unseen Scenarios

Method	Points2Plans	Points2Plans–Delta
Unseen sizes of objects	**58%**	33%
Unseen camera view angles	**61%**	42%
Unseen environments	**50%**	33%
Unseen shapes of objects (YCB objects)	**58%**	49%
Unseen poses of objects	**69%**	51%

We show the extra simulation success rates of Points2Plans and the best-performing baseline to demonstrate the model's generalization ability to unseen camera viewpoints, environments, and different sizes, shapes, and poses of objects. We run 100 trials per approach per generalization metric in the constrained packing task.

From the results shown in the table. 1, we find Points2Plans performs well when it generalizes to unseen scenarios and outperforms the baseline. Please refer to Fig. 10 for the visualizations of comparisons between training datasets and test datasets with novel scenes.

A.13 Generalization to Noisy Segmentation Masks

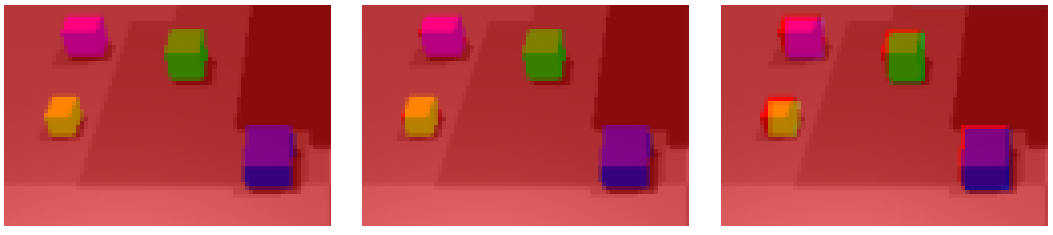

Figure 11: Examples of ground truth segmentation mask and noisy segmentation masks.

We show our method's and the best-performing baseline's robustness to the noise in the segmentation mask. We use the erosion and dilation algorithm [94] to generate the noise for the segmentation masks. We use

Table 2: Robustness to Noisy Segmentation Masks

Method	Points2Plans	Points2Plans−Delta
No Noise	**81%**	69%
Small Noise	**78%**	67%
Large Noise	**73%**	39%

kernel size = 5 for generating small noise and kernel size = 10 for generating large noise to segmentation masks. We run 100 trials per approach per noise metric in the constrained packing task.

From the results shown in the table. 2, we find Points2Plans performs well in the robustness to the noise for the segmentation mask while the baseline performs poorly, especially with large noise. Please refer to Fig. 11 for the details and the visualizations of noisy segmentation masks.