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], and Jeannette Bohg[1]

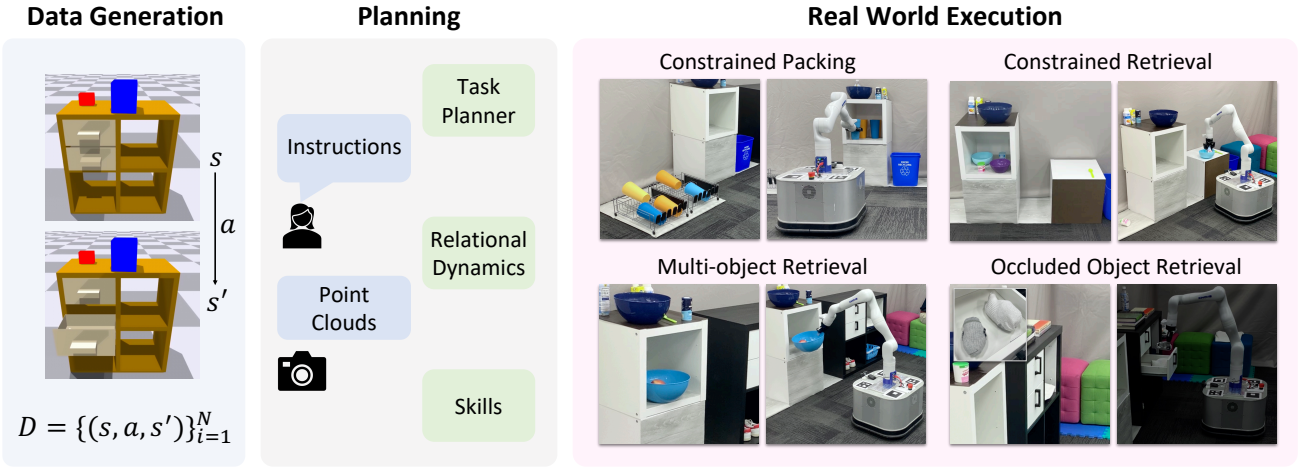

Fig. 1: Training, planning, and execution phases of Points2Plans. Left: In simulation, we sample an environment state and execute a manipulation primitive at random to generate a dataset of single-step environment transitions and train a relational dynamics model. **Middle:** At planning time, Points2Plans 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.

***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%.**

I. 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) [6–8]. 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 [9–11]. 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

[1]Stanford University. [2] University of Utah. [3] Princeton University. [4] NVIDIA Research.

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 $(s_1, a_1, \ldots, s_H)$ 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 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. For qualitative demonstrations of our approach operating on a mobile manipulator platform and supplementary materials, please refer to our project page available at sites.google.com/stanford.edu/points2plans.

II. Related Work

A standard approach to solving **long-horizon manipulation tasks** sequences manipulation *skills* [16–21] according to a high-level plan produced by symbolic planners [2,22–26], language models [1,11,27–30], or combinations [31–35]. 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. 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]. 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 [36–38], parameterized action Markov decision processes [7,39–42], 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 [6,9]. 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 [49–54], but predict state changes at small timescales. Graph neural networks can predict deformable [55,56] and multi-object [57–62] 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.

III. 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., $\texttt{Above}(\mathit{cup}, \mathit{table})$), and a *fact* is an assertion of truth over a ground predicate (e.g., $\texttt{Above}(\mathit{cup}, \mathit{table}) = \text{true}$). Thus, we define the goal state as a conjunction of desired facts $\mathcal{G} = g_1 \wedge \ldots \wedge g_M$, 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 [66,67].

Manipulation Primitives. To solve long-horizon tasks, we assume access to a library of manipulation primitives $\mathcal{L}^\phi = \{\phi^1, \ldots, \phi^K\}$. 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

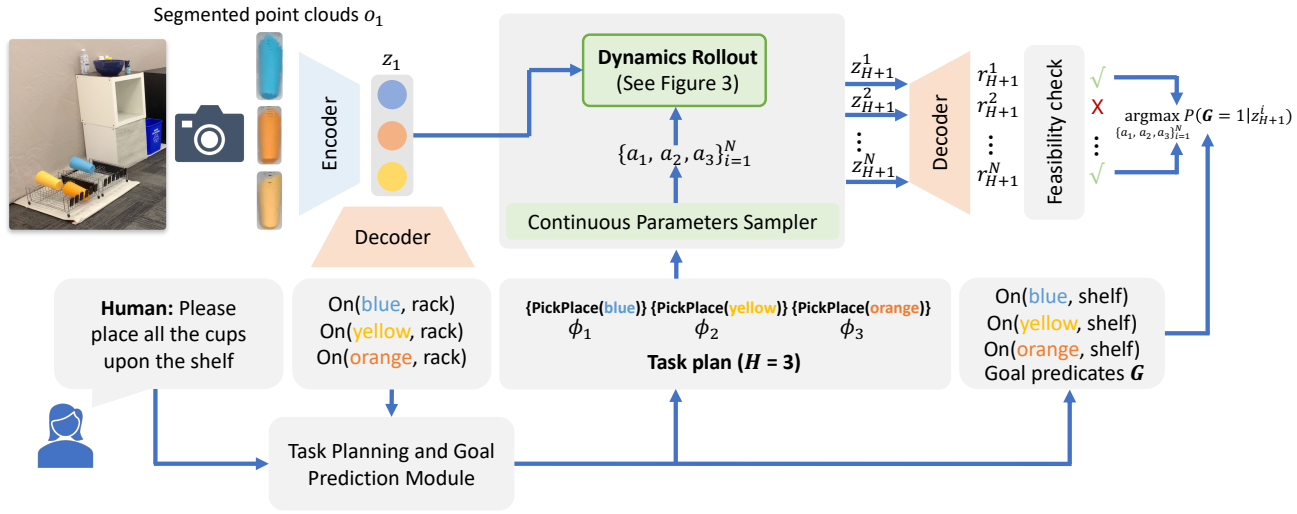

Fig. 2: Overview of Points2Plans. A partial-view segmented point cloud $\mathbf{o}_1$ is first encoded into the (object-centric) latent state $\mathbf{z}_1$. The latent state $\mathbf{z}_1$ is then decoded into predicates that serve as environment context for the task planning and goal prediction module (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.

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 [68] and detection [69].

The Planning Objective. Given an instruction l and segmented partial-view point clouds $\mathbf{o}_1$, our objective is to compute a plan $\tau = [\psi_1, \ldots, \psi_H]$ (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:

$$\underset{\mathcal{G}, \psi_{1:H}}{\arg\max} \quad p\left(l \mid \mathcal{G}, \mathbf{o}_1\right) p\left(\mathcal{G} \mid \psi_{1:H}, \mathbf{o}_1\right). \tag{1}$$

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 $\psi_{1:H}$.

IV. Proposed Approach: Points2Plans

The planning objective in Eq. 1 can be optimized via a hierarchical approach [70] 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

$$\underset{G, \psi_{1:H}}{\arg\max}\ q_1\left(\phi_{1:H}, \mathcal{G} \mid l, \mathbf{o}_1\right) q_2\left(a_{1:H} \mid \phi_{1:H}, \mathcal{G}, \mathbf{o}_1\right). \tag{2}$$

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 [71] 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. IV-A), a hybrid rollout strategy for predicting point cloud states (Sec. IV-B), and finally, we present our full planning approach (Sec. IV-C).

A. 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 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,

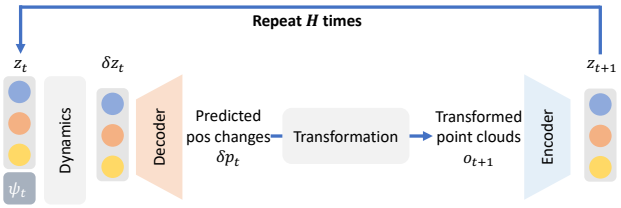

Fig. 3: Points2Plans hybrid rollout strategy.

a transformer-based dynamics model T, and a decoder Dec, all of which are jointly trained on single-step environment transitions. We describe the details of each component below.

Encoder. The encoder Enc takes as input segmented point clouds $\mathbf{o}_t = o_t^1, \ldots, o_t^M$ at timestep t and produces a factored, object-centric latent state $\mathbf{z}_t = z_t^1, \ldots, z_t^M$, where M is the number of objects in the scene (which may vary across tasks). It embeds each per object segment using PointConv [72] and appends a learned positional embedding in PyTorch [73] 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. V that the choice of delta dynamics translates to notable improvements in pose and predicate prediction accuracy.

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 $\mathbf{r}_t = Dec_r(\mathbf{z}_t) = \{ p(u = \mathrm{true} \mid \mathbf{z}_t) \mid \forall u \in \mathcal{U} \}$. The probability $p(u = \mathrm{true} \mid \mathbf{z}_t)$ 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.

B. 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. IV-A) 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.

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.

C. 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 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 $\{\tilde{\mathcal{G}}^i\}_{i=1}^N$. For each task plan $\tilde{\phi}_{1:H}$ and goal $\tilde{\mathcal{G}}$ predicted by the LLM, we 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:

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

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

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

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

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

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

We optimize Eq. 3 to maximize the probability of achieving goal predicates $\tilde{\mathcal{G}}$ using sampling-based optimization techniques [74]. 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 $Dec_r^g(\mathbf{z}_{H+1})$. $\mathcal{C}$ 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 $Dec_r^c(\mathbf{z}_t)$) 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. IV-B).

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 $\prod_{g \in \tilde{\mathcal{G}}} Dec_r^g(\mathbf{z}_{H+1})$ 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.

V. 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 [75] to train our RD model and use GPT-4 [76] 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. IV-B). 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., `Blocking`(a, b).

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.

Across all tasks, we use three manipulation primitives corresponding to pick-and-place, pick-and-toss, and open/close actions. Further details on our experiments (e.g., primitives, predicates, hardware, training, prompts, and implementation details) are provided in the supplementary materials made available at sites.google.com/stanford.edu/points2plans. The supplementary materials also includes extended experiments studying Points2Plans' generalization to novel scenarios and robustness to segmentation noise.

VI. 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 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 shows 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 see that the Greedy approach performs significantly worse than Points2Plans in Fig. 4a, indicating that multi-step planning is required for the long-horizon tasks (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

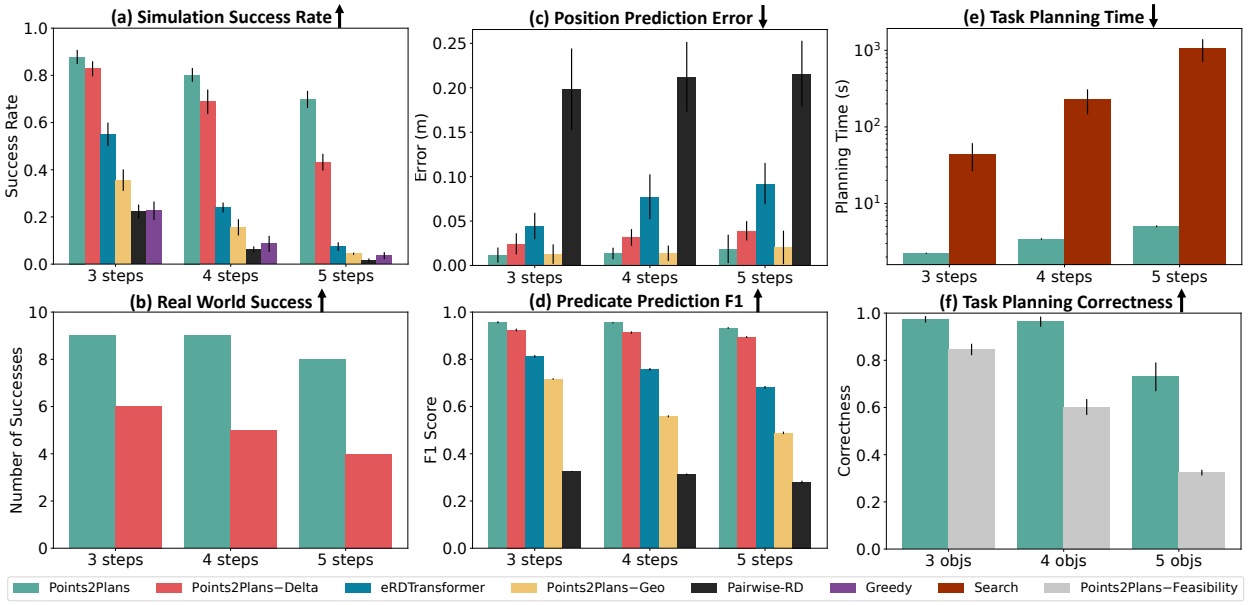

Fig. 4: Simulation and real-world results for the **Constrained Packing** (a-d) and **Constrained Retrieval** (e-f) tasks. As task complexity increases, Points2Plans significantly outperforms baselines in terms of planning success rate (a-b), position prediction error (c), and predicate classification 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.

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).

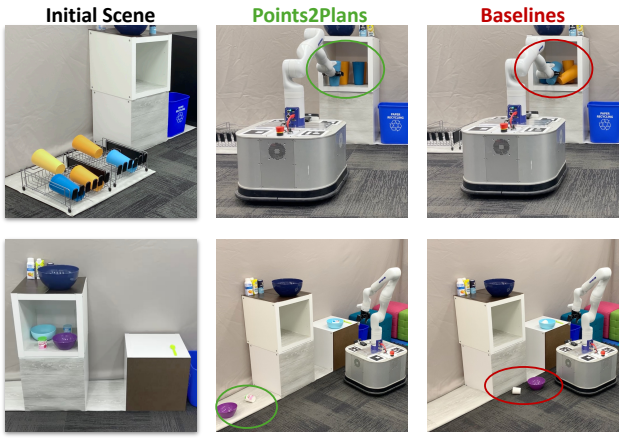

Fig. 5: Points2Plans generalizes to unseen long-horizon tasks, whereas the baselines struggle to find collision-free plans.

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. The baseline fails to remove occluding objects before attempting to grasp the target objects behind them, while Points2Plans infers a feasible task plan based on feasibility-related predicates detected by the RD model.

VII. 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. Future work includes the design of methods to identify and recover from execution failures, online fine-tuning of the relational dynamics model to improve real-world transfer, and interfacing with closed-loop policies to solve tasks that require finer-grained motions.

VIII. Acknowledgements

This work was partially supported by NSF Awards #2024778 and #2149585, 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.