Planning with Spatial-Temporal Abstraction from Point Clouds for Deformable Object Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Effective planning of long-horizon deformable object manipulation requires suitable abstractions at both the spatial and temporal levels. Previous methods typically either focus on short-horizon tasks or make strong assumptions that full-state information is available, which prevents their use on deformable objects. In this paper, we propose PlAnning with Spatial-Temporal Abstraction (PASTA), which incorporates both spatial abstraction (reasoning about objects and their relations to each other) and temporal abstraction (reasoning over skills instead of low-level actions). Our framework maps high-dimension 3D observations such as point clouds into a set of latent vectors and plans over skill sequences on top of the latent set representation. We show that our method can effectively perform challenging sequential deformable object manipulation tasks in the real world, which require combining multiple tool-use skills such as cutting with a knife, pushing with a pusher, and spreading dough with a roller. Additional materials can be found at our project website.[1]

Keywords: Long-horizon Planning, Deformable Object Manipulation

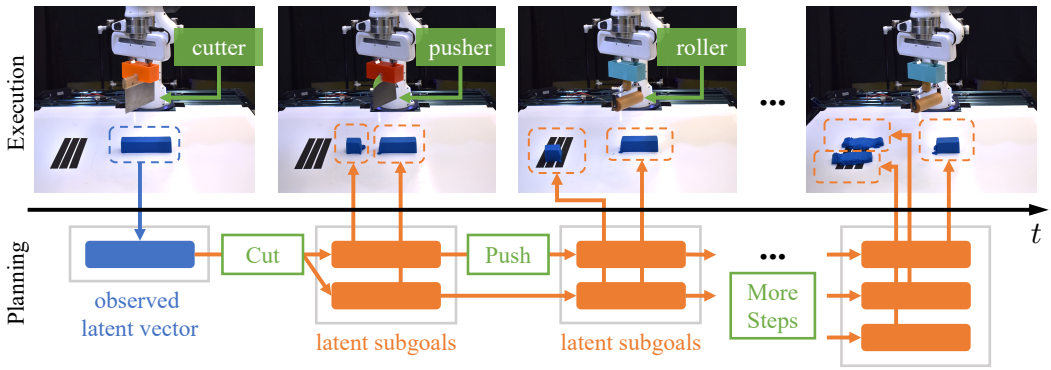

Figure 1: Long-horizon dough manipulation with diverse tools. Our framework is able to solve long-horizon, multi-tool, deformable object manipulation tasks that the agent has not seen during training. The illustrated task here is to cut a piece of dough into two with a cutter, transport the pieces to the spreading area on the left (with a high-friction surface) using a pusher, and then flatten both pieces with a roller.

1 Introduction

Consider a typical cooking task of making dumplings from dough. People plan over which piece of dough to manipulate and which tool to use in sequence, incorporating both spatial and temporal abstractions. A spatial abstraction reasons about objects, parts, and their relations to each other, such as reasoning about pieces of dough instead of reasoning about individual dough atoms; such a

[1] https://sites.google.com/view/pasta-plan

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

spatial abstraction enables efficient planning and compositional generalization. On the other hand, a temporal abstraction incorporates abstract actions represented as a set of skills such as deciding which tool to use at different task stages, instead of making plans at low-level actions such as joint torques. Temporal abstractions allow planning at the skill level, enabling more efficient optimization for solving long-horizon tasks. An autonomous robot that operates in unstructured environments should be able to reason about world dynamics using high-level spatial and temporal abstractions instead of reasoning only over the atoms, infinitesimally small timesteps, and low-level robot actions.

The research community is making rapid progress towards developing state abstractions for manipulating complex objects, including key points [1], graphs [2, 3], dense object descriptors [4], or implicit functions [5, 6]. However, most of these approaches do not make abstractions at the temporal level, limiting their use to short-horizon tasks. Methods are also being developed with temporal abstractions, planning over a set of skills to solve long-horizon tasks [7, 8, 9]. However, the lack of spatial abstraction severely limits their generalization ability. Therefore, it remains a key question in robot learning on how to learn spatial and temporal abstractions within a unified framework for complex and long-horizon manipulation tasks.

In this work, we focus on the challenging task of sequential deformable object manipulation, as shown in Figure 1. We consider a set of dough manipulation tasks that require sequentially applying different skills using multiple tools to manipulate dough, such as spreading using a roller, cutting using a knife and pushing using a pusher, where the longest task requires applying 6 skills in sequence. Deformable objects like dough have nearly infinite degrees of freedom. As such, in this work, we dynamically cluster points in a point cloud into different groups and learn a point cloud encoder to map each element in the group into a latent vector. In this way, we obtain a compositional 3D set representation of the state space. Given an observation and a target point cloud, we then sample skill sequences with subgoals generated in this latent space. We learn skill abstraction modules to determine the feasibility and score of each skill sequence to plan for the skill sequences and the subgoals to reach the goal.

Our contribution of this paper is a framework PASTA, PlAnning with Spatial-Temporal Abstraction, that learns a set of skill abstraction modules over a 3D set representation, which enables our method to efficiently plan over both a spatial and temporal abstraction of the environment. With this spatial-temporal abstraction, PASTA can compose a set of skills to solve complex tasks with more entities and longer-horizon than what was seen during training. We show that PASTA significantly outperforms an ablation that performs planning with a flat representation without a spatial abstraction (e.g. without a 3D set representation). Finally, our planner can be trained in simulation and transferred to the real world for solving sequential tasks of deformable object manipulation.

2 Related Work

Model-based Planning for Sequential Manipulation. Model-based systems for sequential manipulation generally take one of two approaches to construct the dynamics model. The first approach, Task and Motion Planning (TAMP), constructs the model based on physical rules. These systems typically assume known object states and known effects for the action operators [10, 11, 12, 13, 14]. However, it is difficult to estimate states and dynamics for unknown objects, deformable objects, or from partial observations. While recent works have made progress in learning certain components of the system, such as the logical states [15] from high dimensional observations or learning action models [16, 17, 18] from interactions, they still require either known states or known action operators. In contrast, our work learns a 3D set representation as well as the action model on the representation.

Another approach learns dynamics directly from visual observations [2, 19, 20]. Most of these works focus on learning a one-step dynamics model for planning short-horizon tasks. A few works learn the dynamics model over a set of skills and use it for sequential manipulation of rigid objects [21, 22] or deformable objects [7]. However, these works does not use an object-centric representation and thus cannot easily generalize to more complex scenes. In contrast, our framework unifies both temporal

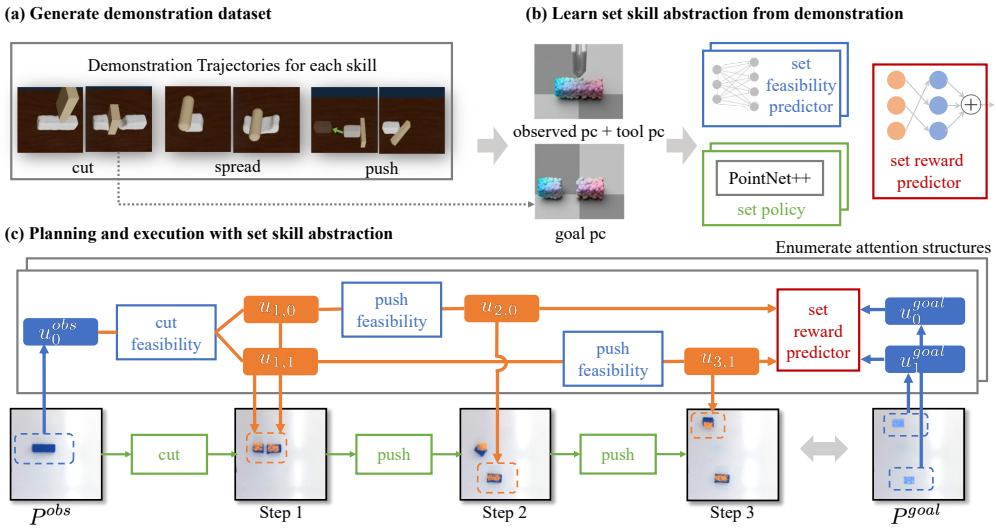

Figure 2: Overview of our proposed framework PASTA. (a) We first generate demonstration trajectories for each skill in a differentiable simulator using different tools. (b) We then sample point clouds (pc) from the demonstration trajectories to train our set skill abstraction modules. (c) We map point clouds into a latent set representation and plan over tool-use skills to perform long-horizon deformable object manipulation tasks. P^{obs}, P^{goal} are the observation and target pc; $u_{i,j}$ denotes component j at step i. The example shows our method performs the CutRearrange task, which requires cutting the dough into two pieces with a knife and transporting each piece to its target location.

and spatial abstraction and can perform long-horizon manipulation for complex tasks with more objects than in previous work, as we will show.

Planning with Spatial Abstraction. Many prior works leverage spatial abstraction to facilitate solving tasks that involve complex dynamics and high-dimensional observations. These works either model a compositional system with Graph Neural Networks (GNN) [2, 3, 6, 23, 24] or learn policies directly from object-centric representations [25, 26]. These works demonstrate compositional generalization, but they only learn policies or one-step dynamics models for planning, which can be difficult for solving long-horizon tasks. In contrast, our framework connects temporally extended spatial abstractions with a feasibility predictor, which allows us to plan over a longer time horizon.

Deformable object manipulation. Deformable objects have nearly infinite degrees of freedom and complex dynamics, making them very challenging to manipulate. Previous works have explored pouring liquid [5, 27, 28], rope manipulation [29, 30], and cloth manipulation [31, 32, 33, 34, 35]. Other papers have also explored manipulating elasto-plastic objects such as deforming them by grasping [2, 36], rolling [37, 38], or cutting [39]. However, these works mostly only consider manipulation with one skill at a time. In contrast, we consider the task of sequential manipulation using multiple tools. The one exception is DiffSkill [7], where multiple skills are chained together. However, DiffSkill uses RGB-D images to represent the scene. In contrast, we use a 3D set representation extracted from point clouds, which enables us to generalize compositionally to tasks with more objects and longer-horizon and also to transfer the planning method to the real world.

3 Method

Given a point cloud of the dough P^{obs} and a goal point cloud P^{goal}, our objective is to execute a sequence of actions $a_1, ..., a_T$ that minimizes the distance between the final observed point cloud and the goal $D(P_T^{obs}, P^{goal})$ where P_i^{obs} is the segmented observation point cloud at time i. We aim to solve long-horizon tasks that require chaining multiple skills in novel scenes with more objects than training. To do so, we present a general framework that incorporates spatial and temporal abstractions for learning and planning with skills from high-dimensional observations, as summarized in Figure 2. We use point clouds as input to all our modules to enable easier transfer from simulation to the real world and to enable robustness to changes in viewpoint.

We assume access to an offline dataset of demonstration trajectories $\mathcal{D}_{demo}$, where each trajectory demonstrates one of the K skills using one tool. We can learn skill policies by imitation learning from these demonstration. To chain these skills to solve long-horizon tasks, we train a set of skill abstraction modules (Sec. 3.2), which can be used for efficient planning in a latent space. Below, we first describe how we generate this latent space and use an object-centric representation to generalize our planner to scenes with more objects.

3.1 Spatial Abstraction from Point Clouds

Scene Decomposition: First, we describe the representation that we use as a spatial abstraction of the point cloud observation. Given a point cloud $P \in \mathbb{R}^{N\times 3}$, we first cluster particles into different components based on their proximity in space. In this paper, we apply DBSCAN [40] to P and group points into a set of entity point clouds $\{P_i \in \mathbb{R}^{N_i\times 3}\}_{i=1\ldots C}$ by separating points from high-density regions into different clusters. While other works on scene decomposition can also be used [41], we find this simple method to be sufficient for our tasks.

Entity Encoding: Planning directly over the $\mathbb{R}^{N\times 3}$ dimensional space of point clouds is too inefficient. Instead, to enable efficient planning in a latent space, we train a point cloud variational autoencoder [42]. The variational autoencoder model includes three modules: A point cloud encoder $\phi : \mathbb{R}^{N\times 3} \to \mathcal{U}$ that maps a point cloud to a latent vector, a decoder $\psi : \mathcal{U} \to \mathbb{R}^{N\times 3}$ that maps from a latent space back into a point cloud, and a prior distribution over the latent space $p_u : \mathcal{U} \to [0, 1]$ which can be used to generate samples from the latent space during planning. We can the encode each entity point cloud P_i into a set of latent vectors: $\{u_i\}_{i=1\ldots C}$ as our set latent representation. We further achieve translation invariance by separating the translation from its shape embedding. See our appendix for details.

3.2 Neural Spatial-Temporal Abstraction

Given a demonstration dataset, we can learn a set of goal-conditioned policies as the skills. However, we assume the demonstration trajectories only show how to perform a single-stage task with a single tool and can be just for a single object, so our skills are also limited to single-stage, single-object tasks. In order to plan with these skills to solve longer-horizon tasks, we further learn a feasibility predictor and a reward predictor. They can be used to plan subgoals that chain the skills into a sequence, such that each subgoal is feasible for the corresponding policy to reach and the final subgoal reaches a given goal. Additionally, all these modules take in the set representation $\{P_1 \ldots P_C\}$ as input such that the planner also generalizes to tasks with more objects.

Set Point Cloud Policy The set point cloud policy for the kth skill π_k takes in an observed point cloud P^{obs}, a goal point cloud P^{goal}, and a tool point cloud P_k^{tool} and outputs an action at each timestep to control the tool directly. The policy only sees a subset of the dough point cloud and goal point cloud in the scene to enable compositional generalization to scenes with more objects. For example, a dough spreading policy will only see the dough being spread. To achieve this, we train each set policy with behavior cloning and hindsight relabeling on the demonstration dataset with an attention mask that filters out the non-relevant entity point clouds. During planning, this attention mask will be provided by the planner.

Set Feasibility Predictor Similar to DiffSkill [7], we train a feasibility predictor $f_k(U^o, U^g)$ for each skill, where $U^o = \{u_i^o\}_{i=1\ldots N_o}, U^g = \{u_j^g\}_{j=1\ldots N_g}$ are latent set representations of an observation and goal point cloud respectively. The feasibility predictor outputs a value in $[0, 1]$ denoting whether the goal can be reached from the observation by executing the kth skill. In DiffSkill, the feasibility predictor uses a flat representation that takes in a single latent vector for all objects in the scene as input. However, as our skills such as cutting or spreading only need to take in a subset of the objects as input, we use the same attention method for the feasibility and assume that the feasibility predictor only takes as input a subset of the full set representation $\hat{U}^o \subseteq U^o, \hat{U}^g \subseteq U^g$, where $\hat{U}^o = \{u_i^o\}_{i=1\ldots N_k}, \hat{U}^g = \{u_j^g\}_{j=1\ldots M_k}$, where N_k and M_k are the number of components in the observation and goal for skill k. The number of components in the observation and goal can be

different since the number of components can change before and after executing a skill; for example, the cut skill takes one component as observation and cuts it into two components. As another example for robot assembly [43], the number of entities increases when a piece is disassembled into parts and the number of entities decreases when the parts are assembled. In this work, we manually define the number of entities N_k, M_k per skill. Determining which subset to attend to when executing each skill can be difficult; we make this decision during planning and defer the details to Sec. 3.3. We parameterize f_k to be invariant to permutation of the set by first max-pooling the transformed latent vectors into a single vector for both $\hat{U}^o$, and $\hat{U}^g$ (after filtering the elements that are not attended to) followed by a Multi-Layer Perceptron (MLP).

We train the feasibility predictor of skill k with positive examples $\hat{U}^o, \hat{U}^g$ (examples in which the goal $\hat{U}^g$ can be reached from the observation $\hat{U}^o$ within T timesteps by executing skill k) and negative examples (in which the goal $\hat{U}^g$ cannot be reached from the observation $\hat{U}^o$ using skill k). During training, we obtain positive pairs for the feasibility predictor by sampling two point clouds (P^{obs}, P^{goal}) from the same trajectory in the demonstration set. To find $\hat{U}^o, \hat{U}^g$, we first cluster the observation and goal point clouds into two sets $\{P_i^o\}, \{P_j^g\}$ respectively. Then, we match point clouds in the observation set to those in the goal set by finding the pairs of point clouds that are within a Chamfer distance of ϵ: $\{(P_i^o, P_j^g) \mid D_{Chamfer}(P_i^o, P_j^g) < \epsilon\}$. We then remove these point clouds from the corresponding set, since these are the point clouds that have already been moved to the target location in the goal. We can then encode the remaining point clouds into $\hat{U}^o, \hat{U}^g$ as explained above. We generate hard negative samples by randomly replacing one of the entities in the positive examples; see the appendix for details.

Set Reward Predictor As we do planning in a latent space, we train a set reward predictor as our planning objective which determines how close a plan is to a given goal. The set reward predictor R takes two latent set representation as input U^o, U^g. Since our tasks focus on matching each entity in the observation with one in the goal, we assume they have the same number of components, i.e. $|U^o| = |U^g| = N_r$. To compute the reward, we try to find the matching entity with the minimal matching cost: $R\Big(\{u_i^o\}, \{u_j^g\}\Big) = \arg\max_\sigma \sum_{i=1}^{N_r} r_\theta(u_i^o, u_{\sigma(i)}^g)$, where σ is a permutation and r_θ is a reward prediction network parameterized by an MLP trained to predict the negative Chamfer distance between the latent vectors of the two point clouds. Optimization of the reward computation is done by performing Hungarian matching between the two sets, where the matching cost is the negative of the predicted reward.

Algorithm 1: Planning with Spatial-Temporal Abstraction (PASTA)

Input : Demonstration Dataset D_{demo}, skill horizon T, planning horizon H, modules for neural skill abstraction π_k, f_k, r_θ, Point Cloud VAE with encoder ϕ, decoder ψ, prior p_u

for *each valid skill sequence* $k_1, \dots k_H$ **do**
  for *each valid attention structure* $U_1, \dots U_H$ **do**
    Initialize different latent subgoals $U_1, \dots U_H$ from p_u ;
    Optimize $U_1, \dots U_H$ according to Sec. 3.3 to obtain cost $C(\mathbf{k}, \mathbf{U})$;

Choose skill sequence $\mathbf{k}$, attention structure $\mathbf{U}$ that minimizes $C(\mathbf{k}, \mathbf{U})$;

for $h \leftarrow 0$ **to** $H - 1$ **do**
  Decode the subgoal from U^h using the decoder ψ ;
  Execute policy π_{k_i} following the subgoal ;

3.3 Planning with Set Representation

Given an observation and a goal point cloud P^{obs}, P^{goal}, we can plan subgoals and the sequence of skills using our trained abstraction modules, such that we can use our skills to follow each subgoal sequentially to reach a given target. To do so, we need to optimize for the sequence of skills to apply, the attention for each skill (i.e. find $\hat{U}^o \subseteq U^o$), as well as the latent subgoals for each skill (i.e. the exact value for each latent vector in $\hat{U}^o$). For our most simple approach, we run a three-level nested optimization: In the top-level, we exhaustively search over the combinations of skills to

apply at each step, i.e. $k_1 \ldots, k_H$, where k_h is the index of the skill to apply at the high-level step h. We only keep the sequences that end with the same set cardinality as the goal by ensuring that $\sum_{h=1}^{H} M_{k_h} - N_{k_h} = N_g - N_o$, where M_{k_h} and N_{k_h} are the number of observation and goal components for the skill applied at step h and N_o and N_g are the number of components in the observed and target point clouds.

In the second-level optimization, we search over different attention structures. Assume that we have N_h components before applying skill k_h, i.e. $|U_h^o| = N_h$ and skill k_h takes K_h components as its observation. We can search over all $C_{N_h}^{K_h}$ combinations of attention structures.. For components not considered by the skill, its latent vector will remain the same at step h. Each choice of each skill attention yields an attention structure, as illustrated in Fig. 2(c). Since all latent subgoals u are initialized independently, we can filter out attention structures that are topologically identical. See the appendix for how we do this efficiently.

In the low-level optimization, for each attention structure, we follow the optimization in DiffSkill [7]. We first sample multiple initializations for all components u, where each node is initialized from our generative model. We can then perform gradient descent to further optimize the latent subgoals on the following objective:

$$\arg\min_{\mathbf{k},\mathbf{U}} C(\mathbf{k}, \mathbf{U}) = \prod_{t=0}^{H-1} f_{k_h}(U^h, U^{h+1}) \exp(-R(U^H, U^g)), \tag{1}$$

where $U^h = \{u_{h,1}, \ldots u_{h,N_h}\}$ are the latent subgoals at step h. $U^0 = U^o$ are the observed set while U^g are the goal set. Finally, we can use our policy to execute our plan by following each subgoal. A summary of our method can be found in Algorithm 1.

4 Experiments

Our experiments are categorized into three parts: In Sec. 4.1, we describe the experimental setups and the baselines we consider. In Sec. 4.2 and Sec. 4.3, we show that PASTA outperforms the baselines and ablate different components in our framework. In Sec. 4.4, we demonstrate that PASTA can be effectively transferred to the real world without any fine-tuning.

4.1 Simulation Tasks and Baselines

Environment setups We consider several long-horizon dough manipulation tasks and divide them into two categories. First, we consider the three tasks from Diffskill [7]: LiftSpread, GatherTransport, and CutRearrange. These tasks require the agent to sequentially compose at most two skills to spread, cut or transport the dough. For these tasks, the number of entities in training and testing are the same. We further propose two new generalization tasks: CutRearrangeSpread (CRS) and CRS-Twice, where there are more entities during testing than during training..

Generalization tasks The CRS task provides a number of demonstration trajectories performing one of the three skills: Cutting with a knife, pushing with a pusher, and spreading with a roller. The demonstration of each skill only shows a tool manipulating a single piece of dough. During testing, the agent needs to cut a dough into two, transport one piece to a spreading area with high table friction and then spread it. Generalization to more entities is required during the pushing and the spreading as there will be two entities in the scene , whereas during training of pushing and spreading there was only one entity. Can we do even more generalization? In the CRS-Twice task, we use the same agent trained on the CRS dataset and ask it to cut two pieces of dough from a chunk, transport both of them to a spreading area and spread them both. This is a 6-horizon task with up to 3 entities in the scene, much more complex than the skill demonstrations the agent is trained on. Due to the long-horizon nature of CRS-Twice, we specify the skill skeleton and use receding horizon planning for all the planning-based methods. See appendix for details.

Baselines We consider several baselines in simulation: First, gradient-based trajectory optimizer with oracle information (Traj-Opt), which can solve single-stage tasks for deformable object manipulation as shown in prior works [44]. Second, model-free RL with Soft Actor Critic [45] with RGB-D image

input (SAC-Image). Third, SAC agent that takes in the dough, target dough, and tool point clouds as input, the same as our method (SAC-Point). Fourth is DiffSkill, model-based planning method from Lin *et al.* [7], which takes RGB-D images as input and has no spatial abstraction. The last one is Flat 3D, which extends DiffSkill to use 3D point clouds as input, with a "flat" 3D representation that encodes the whole scene to a single latent vector without any spatial abstraction.

Metric Following DiffSkill, we report the normalized decrease in the Earth Mover Distance (EMD) approximated by the Sinkhorn diverge [46] computed as $s(t) = \frac{s_0 - s_t}{s_0}$, where s_0, s_t are the initial and current EMD. We additionally set a threshold for the score to determine the success of a trial.

4.2 Comparison with Baselines in Simulation

Method \ Task (Horizon)	DiffSkill tasks			Generalization tasks	
	LiftSpread (2)	GatherMove (2)	CutRearrange (3)	CRS (3)	CRS-Twice (6)
Traj-Opt (Oracle) [44]	0.818 / 40%	0.403 / 0%	0.511 / 20%	0.312 / 0%	0.227 / 0%
SAC-Image [45]	0.797 / 0%	0.567 / 20%	0.103 / 0%	0.562 / 0%	0.365 / 0%
SAC-Point [45]	0.796 / 0%	0.603 / 40%	0.147 / 0%	0.573 / 0%	0.353 / 0%
DiffSkill-Image [7]	**0.920 / 100%**	0.683 / 60%	0.249 / 20%	-0.505 / 0%	-
Flat 3D (Ours)	*	*	0.797 / 60%	-0.712 / 0%	-
PASTA (Ours)	0.904 / **100%**	**0.715 / 100%**	**0.837 / 80%**	**0.896 / 100%**	**0.604 / 40%**

Table 1: Normalized improvement and success rate of all methods on two sets of tasks: tasks in DiffSkill and tasks that require generalization to more steps and entities. For CRS and CRS-Twice, training data only contains skills operating on **one** component of dough but at test time there are more than two components. Only the best performing baselines in CRS are evaluated on CRS-Twice. For LiftSpread and GatherMove, we consider the whole scene as a single spatial abstraction, so Flat 3D is equivalent to PASTA.

Table 1 shows the quantitative results of simulation tasks. First, we show that using a 3D representation is beneficial to planning and complex manipulation, as PASTA matches DiffSkill in LiftSpread and outperforms it in all the other tasks. Second, we highlight the compositional generalization power of PASTA in CRS and CRS-Twice, where the testing configuration differs vastly from training. In both environments, there exist additional components of dough in the scene at test time, and in CRS-Twice, the planner needs to optimize a six-step sequence. Effectively using spatial abstraction to model the scene, PASTA achieves 100% success rate in CRS, and retains a good performance in CRS-Twice. All the baselines, especially the planning baselines (DiffSkill, Flat 3D) fail dramatically, as they can only produce plans that consist of scenes seen in training. Impressively, PASTA is the only approach that reaches non-zero success rate on these tasks that require compositional generalization.

4.3 Ablation analysis

Ablation Method	Performance / Success
No Hard Negatives Feasibility	0.740 / 40%
No Sampling Planning	-0.455 / 0%
No Gradient Planning	0.329 / 0%
PASTA (Ours)	**0.837 / 80%**

Table 2: Ablation results from CutRearrange.

Table 2 shows the quantitative performance of each ablation and PASTA in CutRearrange. First, we consider a variant of feasibility predictor's training, which removes the hard negative samples and only uses random negative sampling (No Hard Negatives), which halves the success rate. Second, we consider two variants of the planner, one without gradient-descent (No Gradient Planning) and one without sampling (No Sampling Planning). The results show that both components are crucial to planning. We provide more ablations on the policy and the planner in the appendix.

4.4 Real World Experiments

Figure 3 shows our real world setup. We use a Franka robot with a top-down Azure Kinect camera capturing the RGB-D observation of the workspace. The robot is equipped with a tool station that allows an automatic change of tools. For real world "dough", we use Kinect Sand as a proxy because of its stable physical property. We transfer the feasibility predictor and reward predictor of PASTA directly from simulation and define heuristic controllers for the skills. For evaluation, we first generate a desired target point cloud and then reset the dough to its initial shape and record its point cloud. Next, given the current and the target point cloud, we use the planner to generate a sequence of skills and subgoals and then execute the plan with our controller. Finally, we record the achieved point

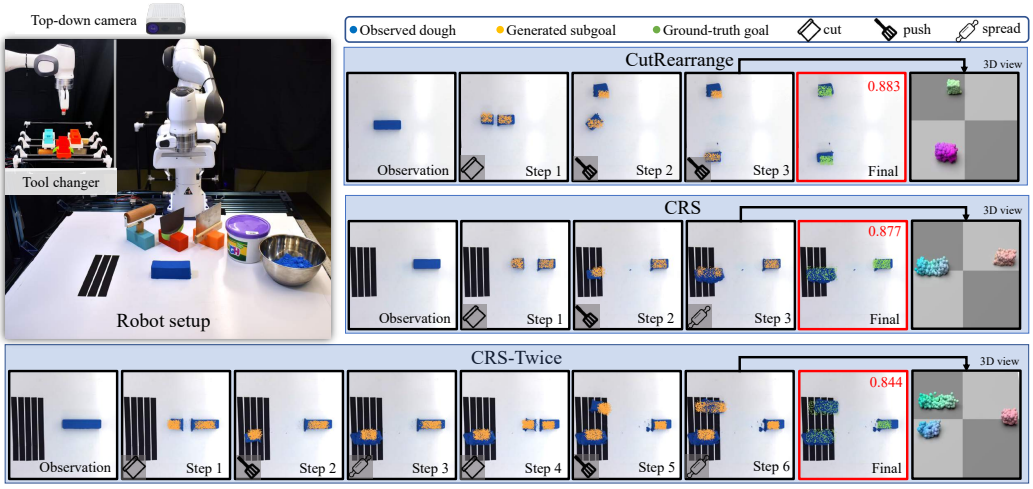

Figure 3: Real world setup and execution with planned subgoals. Our workspace consists of a Franka robot, a top-down camera, and a novel tool changer behind the robot that allows the robot to automatically switch tools. For each task, we show frames after executing a skill overlaid with the decoded point cloud subgoal; we report the final performance in red and overlay the ground truth target in green in the final frame. Additionally, we include a 3D view of the last generated subgoal to show the shape variations.

cloud and report the normalized improvement EMD. We compare with the Flat3D method and also report performance of human on these tasks.

We evaluate on three of the simulation tasks: CutRearrange, CRS, and CRS-Twice. For each task we evaluate on the same four initial and target shapes for each method and report the performance in Table 3. Figure 3 shows the key frames from the execution of PASTA. We overlay the planned subgoals as well as the final goal for qualitative comparison. PASTA performs on-par with human in the real world, highlighting the robustness of our planner and the advantage of using 3D representation for sim2real transfer.

Method \ Task (Horizon)	CutRearrange (3)	CRS (3)	CRS-Twice (6)
Flat 3D	0.351 ± 0.478	0.007 ± 0.429	-
PASTA (Ours)	$\mathbf{0.836} \pm \mathbf{0.029}$	$\mathbf{0.854} \pm \mathbf{0.016}$	$\mathbf{0.795} \pm \mathbf{0.035}$
Human	0.910 ± 0.014	0.863 ± 0.018	0.895 ± 0.013

Table 3: Normalized improvement on real world tasks. Each entry shows the mean and std of the performance over 4 runs. Flat 3D does not produce any meaningful plan for CRS, so we do not evaluate it on CRS-Twice.

5 Conclusions and Limitations

In this work, we propose a planning framework named PASTA that incoporates both spatial and temporal abstraction by planning with a 3D latent set representation with attention structure. We demonstrate a manipulation system in the real world that uses PASTA to plan with multiple tool-use skills to solve the challenging deformable object manipulation tasks, and we show that it significantly outperforms a flat 3D representation, especially when generalizing to more complex tasks.

Limitations: First, we rely on an unsupervised clustering method for entity decomposition and a point cloud VAE for mapping an observation to our latent set representation. This design is specific to the tasks we show and may not generalize to other tasks without retraining or parameter tuning. We hope that our framework can be incorporated in the future with self-supervised methods for learning the spatial abstraction. Second, manipulating deformable objects like dough is very challenging with a significant sim2real gap. This paper provides a starting point for planning with multiple tools towards this challenge; we hope our work can inspire more works in this exciting area.

References

[1] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019.

[2] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

[3] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2021.

[4] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning*, pages 373–385. PMLR, 2018.

[5] 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.

[6] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *International Symposium of Robotics Research*, 2022.

[7] 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. *ICLR*, 2022.

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

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

[10] 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.

[11] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. *Robotics: Science and Systems Foundation*, 2018.

[12] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[13] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. *Technical Report CVC TR-98-003/DCS TR-1165*, 1998.

[14] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[15] 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.

[16] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer. Search-based task planning with learned skill effect models for lifelong robotic manipulation. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[17] E. Ugur and J. Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2627–2633, 2015. doi:10.1109/ICRA.2015.7139553.

[18] 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.

[19] 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.

[20] 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.

[21] K. Fang, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. In *Conference on Robot Learning*, pages 42–52. PMLR, 2020.

[22] 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 *4th Conference on Robot Learning (CoRL 2020)*. Massachusetts Institute of Technology, 2020.

[23] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

[24] X. Ma, D. Hsu, and W. S. Lee. Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, 2021.

[25] N. Heravi, A. Wahid, C. Lynch, P. Florence, T. Armstrong, J. Tompson, P. Sermanet, J. Bohg, and D. Dwibedi. Visuomotor control in multi-object scenes using object-aware representations. *arXiv preprint arXiv:2205.06333*, 2022.

[26] C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7111–7118. IEEE, 2018.

[27] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2629–2636. IEEE, 2017.

[28] N. Gautham, Z. Kai, E. Ben, L. Xingyu, and H. David. Transparent liquid segmentation for robotic pouring. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[29] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

[30] P. Mitrano, D. McConachie, and D. Berenson. Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170, 2021.

[31] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE, 2010. URL https://ieeexplore.ieee.org/abstract/document/5509439.

[32] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg. VisuoSpatial Foresight for Physical Sequential Fabric Manipulation. *Autonomous Robots (AURO) journal*, 2021.

[33] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2021.

[34] Z. Huang, X. Lin, and D. Held. Mesh-based dynamics model with occlusion reasoning for cloth manipulation. *Robotics: Science and Systems (RSS)*, 2022.

[35] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, 2021.

[36] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *Robotics: Science and Systems (RSS)*, 2022.

[37] N. Figueroa, A. L. P. Ureche, and A. Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 611–612. Ieee, 2016.

[38] C. Matl and R. Bajcsy. Deformable elasto-plastic object shaping using an elastic hand and model-based reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3955–3962. IEEE, 2021.

[39] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, July 2021. doi:10.15607/RSS.2021.XVII.067.

[40] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[41] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

[42] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *ICCV*, 2019.

[43] Y. Lee, E. S. Hu, and J. J. Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6343–6349. IEEE, 2021.

[44] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations*, 2020.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[46] T. Séjourné, J. Feydy, F.-X. Vialard, A. Trouvé, and G. Peyré. Sinkhorn divergences for unbalanced optimal transport. *arXiv preprint arXiv:1910.12958*, 2019.