DIFFSKILL: SKILL ABSTRACTION FROM DIFFERENTIABLE PHYSICS FOR DEFORMABLE OBJECT MANIPULATIONS WITH TOOLS

Xingyu Lin*
Carnegie Mellon University
xlin3@andrew.cmu.edu

Zhiao Huang
UC San Diego
z2huang@eng.ucsd.edu

Yunzhu Li
MIT
liyunzhu@mit.edu

Joshua B. Tenenbaum
MIT BCS, CBMM, CSAIL
jbt@mit.edu

David Held
Carnegie Mellon University
dheld@andrew.cmu.edu

Chuang Gan
MIT-IBM Waston AI Lab
chuangg@mit.com

ABSTRACT

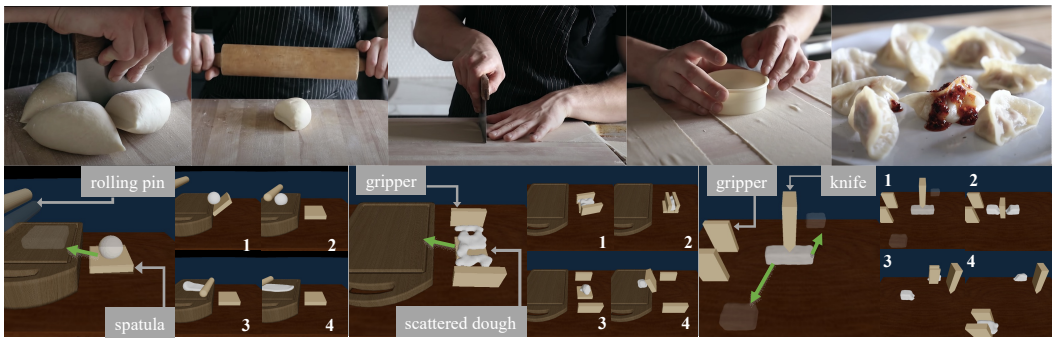
We consider the problem of sequential robotic manipulation of deformable objects using tools. Previous works have shown that differentiable physics simulators provide gradients to the environment state and help trajectory optimization to converge orders of magnitude faster than model-free reinforcement learning algorithms for deformable object manipulation. However, such gradient-based trajectory optimization typically requires access to the full simulator states and can only solve short-horizon, single-skill tasks due to local optima. In this work, we propose a novel framework, named DiffSkill, that uses a differentiable physics simulator for skill abstraction to solve long-horizon deformable object manipulation tasks from sensory observations. In particular, we first obtain short-horizon skills using individual tools from a gradient-based optimizer, using the full state information in a differentiable simulator; we then learn a neural skill abstractor from the demonstration trajectories which takes RGBD images as input. Finally, we plan over the skills by finding the intermediate goals and then solve long-horizon tasks. We show the advantages of our method in a new set of sequential deformable object manipulation tasks compared to previous reinforcement learning algorithms and compared to the trajectory optimizer. Videos are available at our project page[1].

1 INTRODUCTION

Robot manipulation of deformable objects is a fundamental research problem in the field of robotics and AI and has many real-world applications such as laundry making (Maitin-Shepard et al., 2010), cooking (Bollini et al., 2013), and caregiving tasks like assistive dressing, feeding or bed bathing (Erickson et al., 2020). The recent development of differentiable physics simulators for deformable objects has shown promising results for solving soft-body control problems (Hu et al., 2019b; Murthy et al., 2020; Heiden et al., 2021; Huang et al., 2021). These differentiable simulators have facilitated gradient-based trajectory optimizers to find a motion trajectory with much fewer samples, compared with black box optimizers such as CEM or reinforcement learning algorithms (Huang et al., 2021; Heiden et al., 2021; Geilinger et al., 2020).

However, the usage of these simulators is often limited in two ways. First, most differentiable physics simulators are only applied to solve short-horizon, single-skill tasks, such as cutting (Heiden et al., 2021), ball throwing (Geilinger et al., 2020), or locomotion (Hu et al., 2019b). This is partly because the gradient only provides local information, and thus, gradient-based optimizers often get stuck in local optima, preventing them from solving long horizon tasks. For example, consider a two-stage task of first gathering scattered dough on a table and then putting it on the cutting board using a spatula (Figure 1). If the objective function is to minimize the distance between all the dough

*This work was done during an internship at the MIT-IBM Watson AI Lab.
[1] https://xingyu-lin.github.io/diffskill/



Task 1: LiftSpread Task 2: GatherTransport Task 3: CutRearrange

Figure 1: Humans use various tools to manipulate deformable objects much more effectively than state-of-the-art robotic systems. This work aims to narrow the gap and develop a method named DiffSkill that learns to use tools like a rolling pin, spatula, knife, etc., to accomplish complicated dough manipulation tasks. Our method learns skill abstraction using a differentiable physics simulator, composes the skills for long-horizon manipulation of the dough, and evalulated in three challenging sequential deformable object manipulations tasks: LiftSpread, GatherTransport, and CutRearrange.

and the cutting board, a gradient-based trajectory optimizer would directly use the spatula to put a small amount of dough on the cutting board without ever using the gathering operation. Second, the agent needs to know the full simulation state and relevant physical parameters in order to perform gradient-based trajectory optimization, which is a great limiting factor for real-world generalization, as reliable full state estimation of deformable objects from sensory data like RGB-D images can often be challenging and sometimes impossible due to ambiguities caused by self-occlusions.

Planning over the temporal abstraction of skills is a powerful approach to tackle the first issue of solving long horizon tasks. However, finding a suitable set of skills is challenging. For example, while standard skills such as grasping an object or moving the robot arm from one pose to another may be manually specified (Toussaint et al., 2018), more complex skills such as cutting or gathering can be difficult to define manually. Therefore, it is desirable to learn these skills automatically.

In this work, we aim to solve a novel set of sequential dough manipulation tasks that involve operating various tools and multiple stages of actions (Figure 1), including lifting and spreading, gathering and transporting, cutting and rearranging the dough. To extend the use of differentiable physics models to these long-horizon tasks and enable the agent to directly consume visual observations, we propose DiffSkill: a novel framework where the agent learns skill abstraction using the differentiable physics model and composes them to accomplish complicated manipulation tasks. Our method consists of three components, (1) a trajectory optimizer that acts as an expert that applies gradient-based optimization on the differentiable simulator to obtain demonstration trajectories, which requires the full state information from the simulator (2) a neural skill abstractor that is instantiated as a goal-conditioned policy, taking RGBD images as input and imitating the demonstration trajectories, and (3) a planner that learns to assemble the abstracted skills by opgimizing the intermediate goals to reach and solve the long horizon tasks. Experiments show that our method, operating on the high-dimensional RGB-D images, can successfully accomplish a set of the dough manipulation tasks, which greatly outperforms the model-free RL baselines and the standalone gradient-based trajectory optimizer.

2 METHOD

Our goal is to learn a policy to perform sequential deformable object manipulation using tools from sensory observations. We utilize a differentiable physics simulator during training. Since it is not feasible to directly use a standalone differentiable physics solver to find an optimal solution for long-horizontal tasks, we propose to first learn to abstract primitive skills from this differentiable physics simulator; we then plan on top of these skills to solve long-horizon tasks. An overview of our framework is shown in Figure 2.

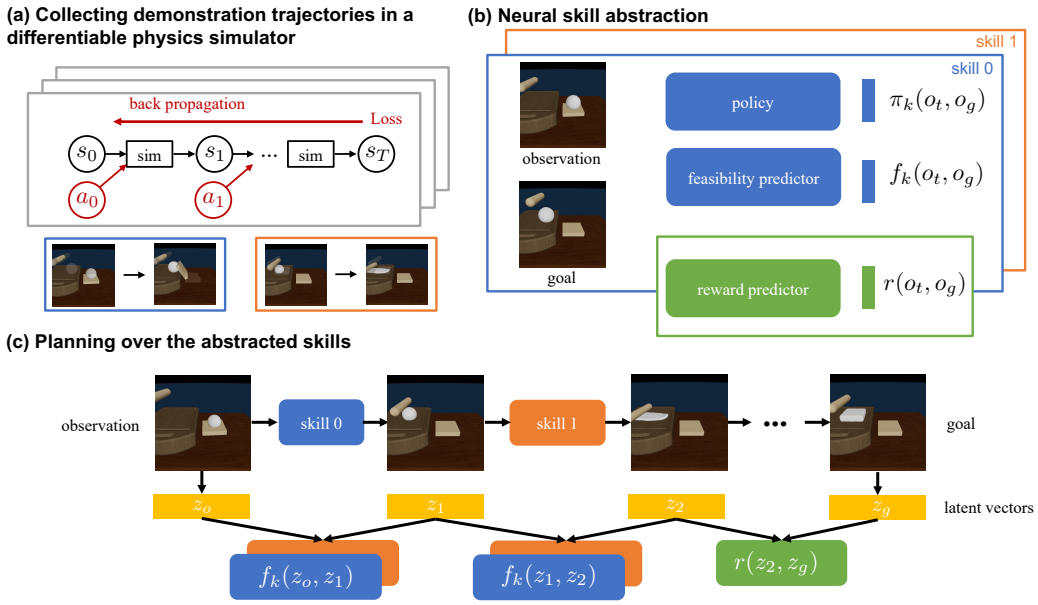


Figure 2: (a) Collecting demonstration trajectories by running a gradient-based trajectory optimizer in a differentiable simulator. (b) Neural abstraction by imitating the expert demonstration, which consist of a goal-conditioned policy, a feasibility predictor and a reward predictor. (c) Planning for both skill combination and the intermediate goals to solve long-horizon tasks.

2.1 Problem Formulation

We consider a Markov Decision Process (MDP) defined by a set of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$ and a deterministic, differentiable transition dynamics $s_{t+1} = p(s_t, a_t)$, with t indexing the discrete time. At each timestep, the agent only has access to an observation $o \in \mathcal{O}$ (such as an image) instead of directly observing the state. For any goal state $s_g \in \mathcal{G}$, a distance function from the state s is given as $D(s, s_g)$. The objective is to find a policy $a_t = \pi(o, o_g)$ that minimizes the final distance to the goal $D(s_T, s_g)$, where T is the length of an episode.

2.2 Collecting Demonstration Trajectories with Differentiable Physics

Previous work has shown that differentiable physics solvers can acquire short-horizon skills for deformable object manipulation in tens of iterations (Huang et al., 2021). Inspired by this work, we first collect demonstration trajectories of using each tool to achieve a short-term goal. Given an initial state s_0, a goal state s_g and the transition dynamics p of a differentiable simulator, we use gradient-based trajectory optimization to solve for an open-loop action sequence (Kelley, 1960):

$$\underset{a_0,\dots a_{T-1}}{\arg\min}\ L(a_0,\dots a_{T-1}) = \underset{a_0,\dots a_{T-1}}{\arg\min} \sum_{t=1}^{T} D(s_t, s_g) + \lambda \sum_{t=1}^{T} E(m(s_t), d(s_t)),\ \text{where } s_{t+1} = p(s_t, a_t), \tag{1}$$

In the case of deformable object manipulation, we represent the current and target shape of the deformable object as two sets of particles and use the Earth Mover Distance (EMD) between the two particle sets as the distance metric $D(s_t, s_g)$, which we approximate with the Sinkhorn Divergence (Séjourné et al., 2019) . To encourage the manipulator to approach the deformable object, we additionally add into the objective the Euclidean distance between the manipulator and the deformable object $E(m(s_t), d(s_t))$, with a weight λ, where $m(s_t)$ and $d(s_t)$ are the positions of the center of mass of the manipulator and deformable object at time t, respectively. We solve Equation 1 by updating the action sequence using $\nabla_{a_t} L, t = 0 \dots T$, with an Adam optimizer (Kingma & Ba, 2014), initialized using an action sequence of all zeros.

In this paper, we define a "skill" as a policy that uses a single tool to achieve a short-horizon goal s_g, starting from an initial state s_0. Each skill corresponds to the use of one tool and can be applied to different observations. For example, in the LiftSpread task, the skill of using the rolling pin can be

applied to either when the dough is on the right or when the dough is on the cutting board. Assuming that we have K tools, the action space over all tools can be written as $[\mathcal{A}^{(1)}, \dots \mathcal{A}^{(K)}]$, where $\mathcal{A}^{(k)}$ is the action space of the k^{th} tool. When collecting demonstration for learning the skills, for each short-horizon goal s_g, we run the trajectory optimizer for each tool separately, by masking the actions for other tools to be zero at each timestep.

2.3 Neural Skill Abstraction

Although the trajectory optimizer is able to provide solutions for short-horizon tasks, it is unable to solve long-horizon tasks due to local optima. Furthermore, running the trajectory optimizer requires knowing the full state of the environment, including particle positions and mass distributions of the deformable objects and the physical parameters of these objects. This information is difficult to obtain during real-world deployment, where the observations available to a robot are from sensory information such as RGB-D images. Additionally, the trajectory optimizer takes minutes to run, which is too slow during evaluation for real-time applications.

As such, we propose to learn a neural skill abstractor that learns skills from the demonstration videos of a trajectory optimizer; we will then leverage these skills for solving long horizon tasks. Our neural skill abstraction consists of a goal-conditioned policy that takes a sensory observation (RGB-D images in our case) as input, a feasibility and reward predictor, as well as a variational auto-encoder (VAE).

Goal conditioned policy: For each tool, we learn a goal-conditioned neural policy $a_t = \pi_k(o_t, o_g) \in \mathcal{A}^{(k)}$, using behavior cloning from the corresponding demonstration trajectories collected for the k^{th} tool. Given the demonstration trajectory $(s_0, a_0, \dots, s_T)$ and the corresponding RGB-D sensory observations $(o_0, \dots o_T)$, as well as an observation of the goal o_g, we train a policy using the MSE loss $L_{policy} = ||\pi(o_t, o_g) - a_t||_2^2$. Furthermore, to make the policy more robust to goal observations outside of the training goal images, we adopt hindsight relabeling (Andrychowicz et al., 2017). When sampling from the demonstration buffer, with a probability p_{hind}, we will relabel the original goal o_g with a hindsight goal $\bar{o}_i$, where $\bar{o}_i \sim \text{Uniform}\{o_{t+1}, \dots o_T\}$. The intuition is that, for any goal that will be achieved by the current action sequence, the policy should imitate the current action.

Variational Auto-Encoder: As will be explained in Section 2.4, we will plan to compose skills in a latent space instead of optimizing directly in the image space. To learn the latent space, we train a generative model of the observation space, specifically a VAE Kingma & Welling (2013). The VAE includes an encoder $z = Q(o)$ that encodes an observation into a fixed length latent vector, and a decoder $o = G(z)$ that decodes back into the observation space. The VAE is trained such that $G(z), z \sim \mathcal{N}(\mathbf{0}, I)$ reproduces the observation distribution.

Feasibility Predictor: For each skill, we learn a feasibility predictor $f_k(z_t, z_g) \in \mathbb{R}$ that classifies if the goal z_g can be reached from the z_t, where $z_t = Q(o_t), z_g = Q(o_g)$. For training the feasibility predictor, we obtain positive pairs by sampling (z_{t_1}, z_{t_2}) from the same demonstration trajectory and negative pairs by sampling pairs of observations from different trajectories. We additionally add negative pairs (z_1, z_2) where both z_1 and z_2 are sampled from a unit Gaussian distribution in the VAE latent space. We assign a label of 1 for positive pairs and a label of 0 for negative pairs. We use an MSE loss L_{fea} for model training, which was shown empirically to work better than a cross-entropy loss. While there may be false negative pairs, we find the feasibility predictor to perform reasonably well, as different trajectories have different initial and goal configurations and it is less likely that the state of a different configuration (e.g. mass of the dough) can be achieved from the current state within one skill. This intuition was similarly used for goal relabeling in previous work by Lin et al. (2019).

Reward Predictor: We further train a reward predictor $r(o_t, o_g) \in \mathbb{R}$ that predicts the negative of the Sinkhorn divergence between the corresponding states $-D(s_t, s_g)$ using an MSE loss L_r. The reward predictor does not depend on any specific skill or tool.

Training We first pretrain the VAE and then freeze the weights. We then jointly optimize the feasibility predictor, reward predictor and the goal conditioned policy.

2.4 Long-horizon Planning with Skill Abstraction

To apply the skill abstraction learned above sequentially, we need to determine 1) which skill to use at each stage; 2) what intermediate goal to specify for each skill. Given o_0, o_g the initial and goal

observations, we plan over H steps. By using the feasibility and reward predictor, we formulate our problem as a hybrid discrete and continuous optimization problem:

$$\underset{k_1,z_1,\ldots,k_H,z_H}{\arg\min} \quad C(\mathbf{k},\mathbf{z}) = -\prod_{i=0}^{H-1} f_{k_i}(z_i, z_{i+1})\bar{r}(z_H, z_g), \text{ s.t. } ||z_i||_2^2 \leq M, \ \forall i. \tag{2}$$

Here, k_i is the index of the tool used at step i, f is the feasibility predictor and $\bar{r}(z_H, z_g) = \exp(-D(z_H, z_g))$ which can be computed from the reward predictor, and z_0, z_g are the VAE encoded latent vectors of the initial and goal observations (RGB-D image) o_0, o_g. The optimization variables include the discrete variables $\mathbf{k} = k_1, \ldots, k_H$, representing the index of the skills to use at each step and the continuous variables $\mathbf{z} = z_1, \ldots, z_H$ that are latent vectors that represent the intermediate goals. The term $||z_i||_2^2$ in the constraint is proportional to the log likelihood of the latent vectors under a unit normal distribution and M is a threshold to ensure that the latent vectors correspond to actual intermediate goals in the observation space.

To optimize for both the discrete and continuous variables in Eqn. 2, we use exhaustive search over all possible combinations of the discrete variables. For the continuous variables, we start with N initial solutions $\{z_1, \ldots z_H\}_j, j = 1, \ldots, N$ and use Adam with projected gradient on the loss for all the initial solutions in parallel, since the constraint is convex. Specifically, after each gradient update step of Adam, we project the current z_i to the constraint set by setting $z_i = \frac{z_i}{\max(||z_i||_2/\sqrt{M}),1)}$. Once we have solved for $k_1, z_1, \ldots, k_H, z_H$, we can decode the latent vectors back to images $o_1, \ldots, o_H = G(z_1), \ldots, G(z_H)$, and call the corresponding goal-conditioned policies sequentially: $\pi_{k_1}(o_0, o_1), \ldots, \pi_{k_H}(\cdot, o_H)$. To simplify the execution, we move each tool to its initial pose at the beginning of executing each skill. The pseudocode is shown in Algorithm 1.

Algorithm 1: Solve long-horizon planning with DiffSkill

Input : Trajectory optimizer, skill horizon T, planning horizon H
Initialize modules for neural skill abstraction π_k, f_k, r, G, Q ;
Generate N demonstration trajectories in differentiable physics $\tau = \{(o_0, a_0, r_0, \ldots, o_T, r_T, o_g)\}_{i=1,\ldots,N}$;
Train neural skill abstraction π_k, f_k, r, G, Q using loss L_{skill} until convergence;
for $\mathbf{k} \in \{0 \ldots K\}^H$ **do**
  Initialize $\mathbf{z} = [z_1, \ldots z_H] \sim \mathcal{N}(\mathbf{0}, I)$;
  Optimize $z_1, \ldots z_H$ according to Eqn. 2 to obtain cost $C(\mathbf{k}, \mathbf{z})$;
Choose $\mathbf{k}, \mathbf{z}$ that minimizes $C(\mathbf{k}, \mathbf{z})$;
for $i \leftarrow 0$ **to** H **do**
  Reset tools to initial poses ;
  Decode intermediate goal images: $o_{g,i} \leftarrow G(z_i)$;
  Execute policy $\pi_{k_i}(\cdot,\ o_{g,i})$ in the environment for T steps ;

3 Experiments

In this section, we will discuss our experimental setup, implementation details and comparison results.

3.1 Experimental setup

Tasks and environments We experiment with a set of sequential deformable object manipulation tasks with dough. We build our simulation environments on top of PlasticineLab (Huang et al., 2021), a differentiable physics benchmark using the DiffTaichi system (Hu et al., 2019a) that could simulate plasticine-like objects based on the MLS-MPM algorithm (Hu et al., 2018). Inspired by the dumpling making process, we design three novel tasks that require long-horizon planning and usage of multiple tools:

- **LiftSpread**: The agent needs to first use a spatula (modeled as a thin surface) to lift a dough onto the cutting board and then adopt a rolling pin to roll over the dough to flatten it. The rolling pin is simulated as a 3-Dof capsule that can rotate along the long axis and the vertical axis and translate along the vertical axis to press the dough.
- **GatherTransport**: Initially, residual of dough is scattered over the table. First, the agent needs to gather the dough with an extended parallel gripper and place it on top of a spatula. Then the agent

needs to use the spatula to transport the dough onto the cutting board. The spatula can translate and rotate along the vertical plane. The gripper can translate along the horizontal plane, rotate around its center and open or close the gripper.
- **CutRearrange**: This is a three-step task. Given an initial pile of dough, the agent needs to first cut the dough in half using a knife. Inspired by the recent cutting simulation (Heiden et al., 2021), we model the knife using a thin surface as the body and a prism as the blade. Next, the agent needs to use the gripper to transport each piece of the cut dough to target locations.

A visualization of these tasks can be found in Figure 1. For all the taks, the agent receives RGB-D image resized to 64x64 from a camera and return velocity control commands directly on the tools.

Evaluation metric We report the normalize decrease in Sinkhorn diverge computed as $s(t) = \frac{s_0 - s_t}{s_0}$, where s_0, s_t are the initial and current Sinkhorn divergence. In this way, a normalized performance of 0 representing a policy that does nothing and a normalized performance of 1 representing an upper bound of a policy that perfectly match the two distributions. The maximum normalized performance of 1 may not always be possible. For example, due to the incompressibility of the dough, certain target shape may be too small for a large dough to fit into. As such, we additionally set a threshold on the Sinkhorn divergence for each task such that the task is assume to be completed successfully when EMD is below the threshold. This threshold is manually picked by observing the performance gap between successful and failed trajectories. For each task, 5 initial and target configurations are given for evaluation, which all require multi-stage manipulation of the dough. We report both the normalized performance metric and the success rate for comparisons.

3.2 Implementation Details of DiffSkill

For each task, we first generate 1000 different initial and goal configurations, varying the initial and target shape of the dough as well as poses of the manipulators. For each configuration, we run the optimizer for each tool separately to generate trajectories of length 50. The Adam optimizer starts with a initial action sequence of all zero and is then run for 200 iterations to optimize each trajectory.

We then train our VAE, policy, feasibility and score predictors over this demonstration video dataset. These modules share the same encoder architecture: 4 convolutional layers with a kernel size of 4 and a stride of 2 and a channel size of $(32, 64, 128, 256)$, followed by an MLP that maps the feature into a fixed 8 dimension vector. The VAE, feasibility and the score predictor also share the same weights for the encoder.

After training, we find the feasibility and score predictor to perform well on the held out trajectories, achieving a L2 error of less than 0.05 for the score predictor and an accuracy of over 0.95 for the feasibility trajectory, across different environments. We found behavior cloning to be sufficient for learning short-horizon skills from the demonstration dataset. In Table 3, we can see that the learned skills (labeled as Behavior Cloning) approach the normalized performance of the trajectory optimization (Trajectory Opt) on single-tool use, although they cannot solve the long-horizon tasks.

Given the skill abstraction, DiffSkill iterates through all skill combination. For each skill combination, we randomly sample 1000 initial solutions for the latent vectors $z_1, \ldots, z_H$, and then perform 500 iterations of Adam optimization for each of them. Since we are optimizing in the latent space, and with the help of parallel computation of GPU, this optimization can be done efficiently in around 10 seconds for each skill combination on a single NVIDIA 2080Ti GPU.

3.3 Baselines

We compare with three strong baselines:

Model-free Reinforcement Learning (RL) We compare with two model-free RL methods: TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018). The RL methods use the same encoder architecture as DiffSkill and receive the negative of the cost function used for the trajectory optimizer as the reward function. We train the RL agents with either a single tool, or with both tools where the agent controls them simultaneously.

Behavior Cloning We compare with another baseline that directly trains a goal-conditioned policy with Behavior Cloning (BC) and hindsight relabeling using all tools. The agent is trained with the same dataset and jointly controls all tools.

Trajectory Opt (Oracle) We compare with the trajectory optimizer used to generate the demonstration, with the same parameters. Note that this method requires full state of the simulation and multiple forward and backward passes through the simulator during evaluation time.

Since DiffSkill sequentially apply multiple skills, we move any used tools to its initial pose after executing each skill with a manual policy.

3.4 Result Analysis

We show that DiffSkill is able to solve the challenging long-horizon, tool-use tasks from the sensory observation (RGB-D) while the baselines cannot. The quantitative results can be found in Table 1. Tool A refers to the spatula for LiftSpread, the parallel gripper for GatherTransport and the knife for the CutRearrange environment, while tool B refers to the rolling pin, the spatula and the gripper for each task respectively.

First, we can compare the rows for single-tool use. All baselines are able to solve one-stage of the task, achieving a reasonable performance, if the correct tool is used (e.g. tool A for LiftSpread and GatherTransport), although for cutting, the performance is still close to zero with cutting only, since cutting itself does not reduce the distance between the current dough and the desired dough locations. BC can well approach the performance of the corresponding trajectory optimizer despite being a policy that only receives RGB-D input. SAC can also solve some of the single-stage tasks such as lifting or transporting, while TD3 performs worse.

Second, with multiple tools, we can see that DiffSkill significantly outperforms the single-skill policy. Remarkably, DiffSkill even outperforms the trajectory optimizer that controls both tools at the same time, which uses the full simulation state during evaluation time. This indicates that high-level planning over skills is necessary for achieving these sequential manipulation tasks. Furthermore, we show visualization of the found solutions of DiffSkill in Figure 3. We can see that DiffSkill is able to find reasonable intermediate goal state such as first putting the dough on the cutting board, or gather dough onto the spatula. CutRearrange is shown to be a harder task, with all methods performing poorly. This is because it requires three stages of manipulation; further, it is non-trivial to transport the dough without deforming it too much. Still, DiffSkill is able to achieve higher success, even though the trajectory optimizer oracle achieves a higher average score. This is because the trajectory optimizer is more reliable at finding partial solutions that transport part of the dough to the target locations but does not complete the full task. Please see our website for visualizations of different methods.

	Method \ Task (H)	LiftSpread (2)	GatherTransport (2)	CutRearrange (3)
Tool A only	Trajectory Opt (Oracle)	0.755 / 0%	0.386 / 0%	0.033 / 0%
	Behavior Cloning	0.754 / 0%	0.356 / 0%	0.018 / 0%
	Model-free RL (TD3)	0.766 / 0%	0.103 / 0%	0.015 / 0%
	Model-free RL (SAC)	0.815 / 0%	0.141 / 0%	-0.004 / 0%
Tool B only	Trajectory Opt (Oracle)	0.092 / 0%	0.494 / 0%	0.239 / 0%
	Behavior Cloning	0.095 / 0%	0.530 / 0 %	0.168 / 0%
	Model-free RL (TD3)	0.001 / 0%	0.000 / 0%	0.121 / 0%
	Model-free RL (SAC)	-0.016 / 0%	0.592 / 20%	0.131 / 0%
Multi-tool	Trajectory Opt (Oracle)	0.818 / 40%	0.403 / 0.%	**0.353** / 0%
	Model-free RL (TD3)	0.781/ 0%	0.423 / 0%	0.133 / 0%
	Model-free RL (SAC)	0.797 / 0%	0.567 / 20%	0.131 / 0%
	DiffSkill (Ours)	**0.920 / 100%**	**0.683 / 60%**	0.297 / **20%**

Table 1: Normalized improvement of all methods and the success rate on different tasks. Each entry shows the normalized improvement / success rate. The top bar shows H, the planning horizon for each environment.

3.5 Ablation Analysis

We perform two ablations on DiffSkill. First, we try removing the planning over the discrete variables that decides which tool to use at each step. Instead, we train a tool-independent policy, the feasibility and score predictor, where the action space is the joint-tool action space. Then during planning, we only need to optimize for the intermediate goals. This ablation is labeled as *No Discrete Planning*.

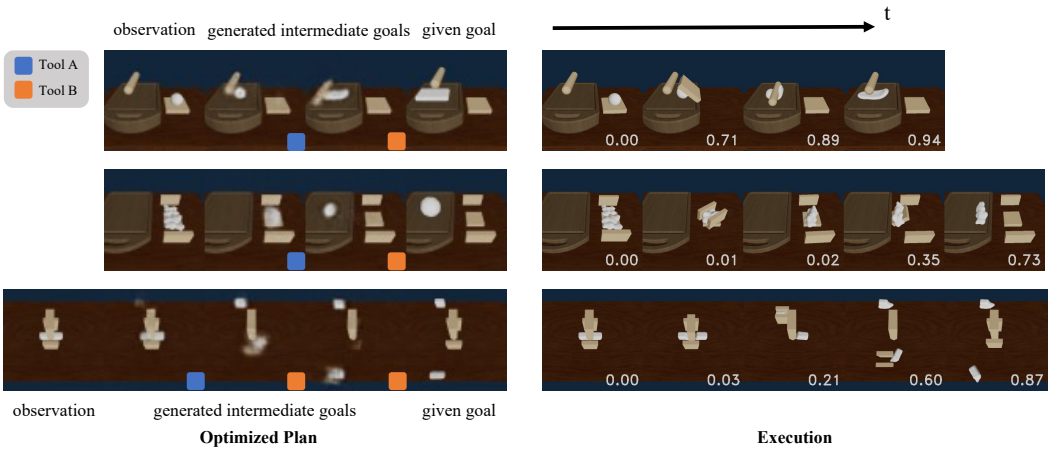


Figure 3: Visualization of the generated plan and the corresponding execution. The plan generated by DiffSkill is shown in the left, where the first and the last image are the given initial and goal observation and in between are the generated intermediate goals. The color blocks indicate which tool is needed to reach the sub-goal. The right shows sampled frames during the execution of the generated plan using the corresponding goal conditioned policy. The numbers on the bottom right shows the achieved normalized improvement metric at that time.

Second, we try to remove the planning over the continuous variables, i.e. the intermediate goals. Without the intermediate goals, we directly use the final target image as the goal for sequentially executing each skill. This ablation is labeled as Direct Execution. Since we do not have the intermediate goals anymore, we try two different ways of choosing the skills to execute at each stage: Randomly pick one skill and an oracle that execute each combination of skills in the simulator and choose the best one in hindsight. The results are shown in Table 2. Without discrete planning, the policy performs poorly. This is probably because during training, a single policy and feasibility predictor is used for learning two modes of skills that are very different and the policy is unable to differentiable for different modes or decide when to switch modes. On the other hand, if we do not optimize for the intermediate goals, we also cannot determine which tools to use at evaluation time, since both the feasibility predictor requires intermediate goals as input. In this case, we can see that using random skills at each stage results in poor performance. Even if the oracle skill order is used, there is still a drop in performance as the policy may not work well given only a future goal instead of an immediate goal.

Method \ Task	LiftSpread	GatherTransport	CutRearrange
No Discrete Planning	0.758 / 20%	0.312 / 0%	0.118 / 0%
Direct Execution (Random)	0.593 / 15%	0.369 / 0%	0.018 / 2.5%
Direct Execution (Oracle)	0.865 / 60%	0.501 / 0%	**0.334 / 20%**
DiffSkill (Ours)	**0.920 / 100%**	**0.683 / 60%**	0.297 / **20%**

Table 2: Normalized improvement and success rate of ablation methods.

4 Related Work

Motion Planning with Differentiable Physics. Differentiable physics models provide the gradients of the future states with respect to the initial state and the input actions. Compared with black-box dynamics models, planning with differentiable physics models can often make more accurate updates on the action sequences and deliver better sample efficiency (Huang et al., 2021). Over the past few years, researchers have built differentiable physics models from first principles rooted in our understanding of physics for various physical systems, ranging from multi-body systems (Degrave et al., 2019; Tedrake & the Drake Development Team, 2019; de Avila Belbute-Peres et al., 2018; Geilinger et al., 2020), articulated bodies (Werling et al., 2021; Qiao et al., 2021), cloth (Qiao et al., 2020; Liang et al., 2019), fluid (Schenck & Fox, 2018), plasticine (Huang et al., 2021), to soft robots (Hu et al., 2019b;a; Du et al., 2021). They have shown impressive performance in using the

gradient to solve inverse problems like parameter identification and control synthesis. A complement thread of methods tried to relax the assumption that we have to know the full state and the physics equations; instead, they employ deep neural networks to learn the dynamics models directly from observation data and use the gradients from the learned models to aid motion planning or policy learning (Battaglia et al., 2016; Chang et al., 2016; Mrowca et al., 2018; Ummenhofer et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Sanchez-Gonzalez et al., 2018; Li et al., 2019; 2018). Our method differs from prior works in that we use the differentiable physics model for skill abstraction and compose the learned skills for long-horizon manipulation of deformable objects.

Deformable Object Manipulation. Deformable objects have more degrees of freedom and are typically more challenging to manipulate than rigid objects. Extensive work has been done in this area, focusing on different types of deformable objects, including plasticine (Li et al., 2018; Huang et al., 2021), fluid (Li et al., 2021), cloth (Maitin-Shepard et al., 2010; Lin et al., 2021; Weng et al., 2021; Ha & Song, 2021; Wu et al., 2019; Yan et al., 2020; Ganapathi et al., 2020), rope (Nair et al., 2017; Sundaresan et al., 2020), and object piles (Suh & Tedrake, 2020). People have also come up with standard benchmarks to establish a fairer comparison between various algorithms (Lin et al., 2020). Our method aims to manipulate dough using tools similar to how humans would do in a kitchen environment (Figure 1). There are also a series of works on tool-using for object manipulation (Toussaint et al., 2018; Qin et al., 2020; Fang et al., 2020; Xie et al., 2019), but most of them focus on rigid objects, whereas we take a step further and tackle deformable objects.

Solving Long Horizon Tasks. Our work aims to solve long-horizon manipulation tasks; thus, it is also closely related to task and motion planning (TAMP) that contains elements of discrete task planning and continuous motion planning. (Gravot et al., 2005; Kaelbling & Lozano-Pérez, 2010; 2013; Toussaint et al., 2018; Garrett et al., 2021). However, most TAMP approaches assume a fully specified state space, dynamics model as well as a set of pre-defined skills such as grasping, pushing, or placing (Toussaint et al., 2018). Differing from prior works, our method learns a policy over RGB-D inputs and generates motion trajectories and skill abstractions by applying gradient-based trajectory optimization techniques using a differentiable simulator.

Our feasibility predictor also relates to previous works that plan over the temporal abstraction of options (Sutton et al., 1999) or skills. In order to do planning, these approaches define a multi-time model (Precup & Sutton, 1997) or equivalently a skill effect model (Kaelbling & Lozano-Pérez, 2017; Liang et al., 2021) to model the end state after executing a skill. In contrast, we model such dynamics implicitly by predicting the feasibility between two latent vectors which allows direct optimization over the sub-goals in a latent space. The most similar to ours is (Nasiriany et al., 2019), which learns a value function as the implicit dynamics. Our method differs from this work by learning skills from a trajectory optimizer instead of using RL. Additionally, we encode multiple skills in different neural networks and perform discrete optimization over the skills for planning.

5 Conclusions

In this paper, we propose DiffSkill, a novel framework for learning skill abstraction from differentiable physics and compose them to solve long-horizontal deformable object manipulations tasks from sensory observation. We evaluate our model on a series of challenging tasks for sequential dough manipulation using different tools and demonstrate that it outperforms both reinforcement learning and standalone differentiable physics solver. We hope our work will be a first step towards more common application of differentiable simulators beyond solving short-horizon tasks.

There are a few interesting directions for future work. First, currently DiffSkill uses exhaustive search for planning over the discrete space. As the planning horizon and the number of skills grow larger for more complex tasks, exhaustive search quickly becomes infeasible. An interesting direction is to incorporate a heuristic policy or value function for more efficient planning. Second, similar to many other data-driven methods, while neural skill abstraction gives good prediction results on region where a lot of data are available, it performs worse when tested on situations that are more different from training. This can be remedied by either collecting more diverse training data in simulation, for example, under an online reinforcement learning framework, or by using a more structured representation beyond RGBD images, such as using an object-centric representation. Third, we hope to extend our current results to the real world, by using a more transferrable representation such as just a depth map or a point-cloud representation. Finally, we hope to see DiffSkill be applied to other similar tasks, such as those related to cloth manipulation.

ACKNOWLEDGMENTS

We thank Sizhe Li for the initial implementation of the CutRearrange environment. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1849154, IIS-2046491, LG Electronics, MIT-IBM Watson AI Lab and its member company Nexplore, ONR MURI (N00014-13-1-0333), DARPA Machine Common Sense program, ONR (N00014-18-1-2847) and MERL.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017. 4

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016. 9

Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pp. 481–495. Springer, 2013. 1

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016. 9

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31: 7178–7189, 2018. 8

Jonas Degrave, Michiel Hermans, Joni Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, 13:6, 2019. 8

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *arXiv preprint arXiv:2101.05917*, 2021. 8

Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C Karen Liu, and Charles C Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10169–10176. IEEE, 2020. 1

Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020. 9

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018. 6

Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph E. Gonzalez, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Learning dense visual correspondences in simulation to smooth and fold real fabrics. *arXiv preprint arXiv:2003.12698*, 2020. 9

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021. 9

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 1, 8

Fabien Gravot, Stephane Cambon, and Rachid Alami. asymov: a planner that deals with intricate symbolic and geometric problems. In *Robotics Research. The Eleventh International Symposium*, pp. 100–110. Springer, 2005. 9

Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *arXiv preprint arXiv:2105.03655*, 2021. 9

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018. 6

Eric Heiden, Miles Macklin, Yashraj S Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.067. 1, 6

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):150, 2018. 5

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019a. 5, 8

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pp. 6265–6271. IEEE, 2019b. 1, 8

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *International Conference on Learning Representations (ICLR)*, 2021. 1, 3, 5, 8, 9

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 9

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013. 9

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Learning composable models of parameterized skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 886–893. IEEE, 2017. 9

Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960. 3

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 9

Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1205–1211. IEEE, 2019. 9

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. *arXiv preprint arXiv:2107.04004*, 2021. 9

Jacky Liang, Mohit Sharma, Alex LaGrassa, Shivam Vats, Saumya Saxena, and Oliver Kroemer. Search-based task planning with learned skill effect models for lifelong robotic manipulation. *arXiv preprint arXiv:2109.08771*, 2021. 9

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *NeurIPS*, 2019. 8

Xingyu Lin, Harjatin Singh Baweja, and David Held. Reinforcement learning without ground-truth state. *arXiv preprint arXiv:1905.07866*, 2019. 4

Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, 2020. 9

Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2021. 9

Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter 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*, pp. 2308–2315. IEEE, 2010. 1, 9

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. *arXiv preprint arXiv:1806.08047*, 2018. 9

J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2020. 1

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2146–2153. IEEE, 2017. 9

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, 2019. 9

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 9

Doina Precup and Richard S Sutton. Multi-time models for temporally abstract planning. *Advances in neural information processing systems*, 10, 1997. 9

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168*, 2020. 8

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. Efficient differentiable simulation of articulated bodies. In *ICML*, 2021. 8

Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7278–7285. IEEE, 2020. 9

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018. 9

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020. 9

Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pp. 317–335. PMLR, 2018. 8

Thibault Séjourné, Jean Feydy, François-Xavier Vialard, Alain Trouvé, and Gabriel Peyré. Sinkhorn divergences for unbalanced optimal transport. *arXiv preprint arXiv:1910.12958*, 2019. 3

HJ Suh and Russ Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020. 9

Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E. Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. *arXiv preprint arXiv:2003.01835*, 2020. 9

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 9

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu. 8

Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. *Robotics: Science and Systems Foundation*, 2018. 2, 9

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019. 9

Thomas Weng, Sujay Man Bajracharya, Yufei Wang, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *5th Annual Conference on Robot Learning*, 2021. 9

Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C. Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021. 8

Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019. 9

Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019. 9

Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *arXiv preprint arXiv:2003.05436*, 2020. 9

A Implementation Details

A list of the hyperparameters used can be found in Table 3.

Model parameter	Value
dimension of latent space	8
MLP hidden node number	1024
Training parameters	**Value**
learning rate	0.001
batch size	128
optimizer	Adam
beta1	0.9
beta2	0.999
weight decay	0
hindsight relabeling ratio p_{hind}	0.5
Feasibility predictor loss weighting λ_{fea}	10
Reward predictor loss weighting λ_r	1
VAE loss weighting λ_{vae}	1
Trajectory optimizer	
learning rate	0.02
number of iteration	200
episode length T	50
action sequence initialization	all initialized to zero
Planning over continuous variables using Adam	
number of iteration	1000
learning rate	0.5
number of initial seeds	1000
log likelihood constraint M	-4 * H

Table 3: Summary of all hyper-parameters.

B Comparison with model-free RL on single-tool tasks

In this work, we focus on solving long-horizon multi-tool tasks. But our proposed method is not limitied to the multi-tool setup. For the single-tool task, or more specifically the single-skill task, our method can be directly applied without planning over the discrete space of which tool to use. To show this, we show results on two single-tool, single-stage tasks: Lift and Spread, which corresponds to the two sub-tasks in LiftSpread, but with a different initial and target dough shapes. For both environments, only one tool can be used: the spatula for Lift and the rolling pin for Spread. The results are shown in Table 4. As these are two simpler tasks, DiffSkill can solve both tasks. SAC can also solve the simpler lift task, but has some trouble learning the spread task. Visualization of both methods on these environments can be found on our website.

Method \ Task	Lift	Spread
SAC	0.890 / 100 %	-0.049 / 0%
DiffSkill (Ours)	0.904 / 100%	0.495 / 100%

Table 4: Normalized improvement of all methods and the success rate on single-tool tasks. Each entry shows the normalized improvement / success rate.