Article:
Wang, S., Leonetti, M. and Dogar, M. orcid.org/0000-0002-6896-5461 (Accepted: 2025) Goal-Conditioned Model Simplification for 1-D and 2-D Deformable Object Manipulation. IEEE Transactions on Robotics. ISSN 1552-3098 (In Press)

Goal-Conditioned Model Simplification for 1-D and 2-D Deformable Object Manipulation

Shengyin Wang, Matteo Leonetti, Mehmet Dogar

***Abstract*—Motion planning for deformable object manipulation has been a challenge for a long time in robotics due to its high computational cost. In this work, we propose to mitigate this cost by limiting the number of picking points on a deformable object within the action space and simplifying the dynamics model. We do this first by identifying a minimal geometric model that closely approximates the original model at the goal state; specifically, we implement this general approach for 1-D linear deformable objects (e.g., ropes) using a piece-wise line-fitted model, and for 2-D surface deformable objects (e.g., cloth) using a mesh-simplified model. Then a small number of key particles are extracted as the pickable points in the action space which are sufficient to represent and reach the given goal. Additionally, a simplified dynamics model is constructed based on the simplified geometric model, containing much fewer particles and thus being much faster to simulate than the original dynamics model, albeit with some loss of precision. We further refine this model iteratively by adding more details from the actually achieved final state of the original model until a satisfactory trajectory is generated. Extensive simulation experiments are conducted on a set of representative tasks for ropes and cloth, which show a significant decrease in time cost while achieving similar or better trajectory costs. Finally, we establish a closed-loop system of perception, planning, and control with a real robot for cloth folding, which validates the effectiveness of our proposed method.**

***Index Terms*—Deformable object manipulation, motion planning, action space reduction, and model simplification.**

I. Introduction

DEFORMABLE object manipulation (DOM) has received significant interest from robotic researchers in recent years [1]–[4] due to the ubiquitous existence of deformable objects in our daily life and their extensive applications in both domestic and industrial scenarios [5]–[8], such as folding laundry or handling surgical materials. DOM tasks have mostly been studied using model-free learning-based approaches in the literature, including reinforcement learning [9], [10], imitation learning [11], and large vision-language-model-based systems [12]. However, while model-based planning methods are widely used for rigid object manipulation, and provide certain advantages such as training-free generalization to novel tasks and objects, their application to deformable objects has been limited due to the significant computational challenges involved. A key challenge in this field is motion planning, where the objective is to develop systems that can autonomously plan trajectories to manipulate deformable objects into specified shapes, such as side folding a piece of cloth in half, as illustrated on the left side of Fig. 1. Typically, a dynamics model is first built in simulation, often utilizing mass-spring systems [13] for their effectiveness. These models usually comprise thousands of particles, each representing a potential picking point, resulting in high-dimensional state representations and a vast action space. Simulating such models is computationally expensive, and motion planners must often evaluate thousands of trajectories, potentially taking hours to plan a single trajectory [14]. These challenges—large action space, high-dimensional state representation, and computationally expensive dynamics—underscore the complexity of current approaches to motion planning for deformable object manipulation and the need for advancements in this area [15]. In this work, we aim to alleviate this computational burden by reducing the action space and simplifying the dynamics model.

A straightforward approach to reduce the action space is to manually limit the pickable points on the deformable object. For example, controlling the shape of a rope by manipulating only its two ends [16], folding or unfolding cloth by focusing on its visible corners [17]–[19], or removing wrinkles from fabric [20]. However, while effective for specific tasks, these methods do not generalize well to different goals. To address this limitation, we propose extracting key picking points and reducing the action space in a goal-conditioned manner. This involves first identifying a simplified geometric model with a minimal number of elements (such as particles or triangles) that can effectively represent the original object at the goal state. For instance, as shown in the top image in the Geometric Model Simplification block of Fig. 1, a minimal model consisting of six triangles and eight particles provides a good approximation of the goal shape when folded sideways in half. By leveraging this minimal model, we extract the key picking points, significantly reducing the search space and enabling the planner to generate efficient trajectories across various tasks/goals.

While action space reduction helps improve efficiency, the complexity of deformable object motion planning is further exacerbated by the high-dimensional state representation and computational expense of existing dynamics models. To mitigate these challenges, many researchers have explored approximate dynamics models as alternatives to high-fidelity ones, typically designed for specific tasks [21], [22]. For particle based models, a straightforward way to simplify the system is by using fewer particles. For example, as shown in Fig. 2(c-d),

S. Wang and M. Dogar are with the School of Computer Science, University of Leeds, Leeds, LS2 9JT, UK (e-mail: {scswan,m.r.dogar}@leeds.ac.uk).

M. Leonetti is with the Department of Informatics, King's College London, London, WC2R 2LS, UK (e-mail: matteo.leonetti@kcl.ac.uk).

M. Dogar was supported by the UK Engineering and Physical Sciences Research Council [EP/V052659/1].

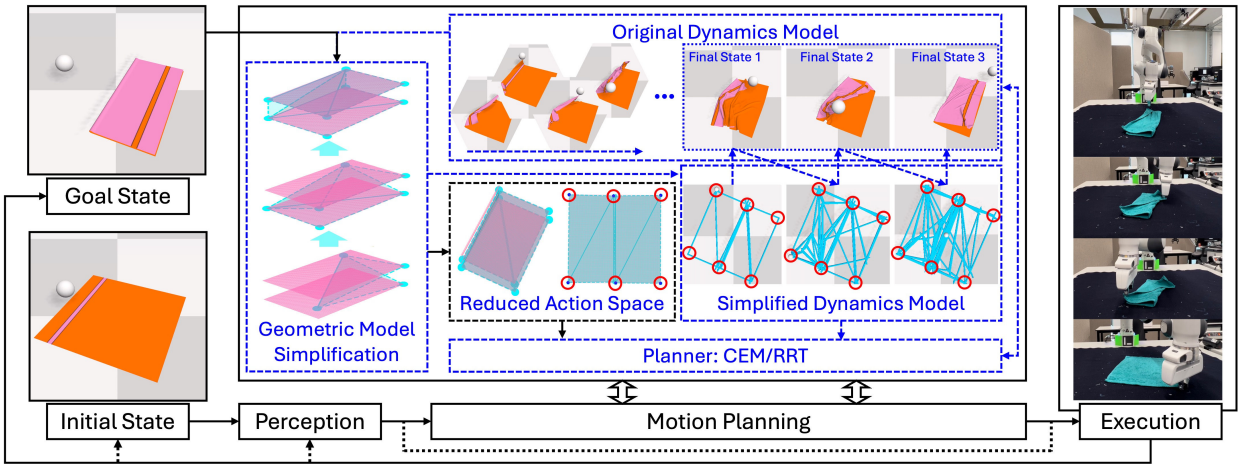


Fig. 1. Overview of the iterative model simplification and motion planning framework for a cloth side folding task, with closed-loop robot execution in the real world. Initially, a simplified geometric model is identified and used to extract key picking points in the reduced action space. A simplified dynamics model is then built and utilized to plan a trajectory in a significantly shorter time. The trajectory is executed on the original model, and if the goal is not reached, the loop iterates, refining the simplified model until a satisfactory trajectory is found. Once a valid trajectory is identified, it is executed on the robot, with the perception system continuously tracking the deformation during manipulation.

coarse grid models can be used as effective approximations for certain tasks, instead of having a dense distribution of particles throughout the cloth. However, a generic grid model is not goal-informed and is only suitable for specific tasks. The grid resolution, which needs to be predetermined by humans, also impacts its effectiveness. To overcome these limitations, we again take a goal-conditioned approach, and propose building simplified dynamics models tailored to specific goals. For instance, the first image in the Simplified Dynamics Model block of Fig. 1 illustrates an approximation of the original cloth dynamics model for a side folding task. Compared to the non-informed grid model, such as Fig. 2(c), the goal-conditioned simplified model has two distinctive halves, with the central edge aligned along the folding line whereas the grid model does not. This model better captures the key dynamics required for side-folding the cloth, and enables much faster trajectory rollouts within the planning framework.

Nonetheless, trajectories planned using simplified dynamics models may not perform well on the original dynamics model due to significant differences between the two. To address this challenge, we propose an iterative framework in which the simplified dynamics model is continuously refined, with each trajectory planned from the previous model serving as a warm start for the next iteration. Our intuition is that if a trajectory planned using a simplified model fails to bring the original model to the goal state, the actual final state achieved can provide valuable information to enhance the simplified model. For example, as shown in the Simplified Dynamics Model block of Fig. 1, while the trajectory planned based on the initial simplified model does not bring the original model to the desired goal state, iteratively refining the model by extracting features from the achieved state can ultimately lead to a successful trajectory. By the third iteration, a trajectory is found that successfully brings the original model to a satisfactory final state. This iterative framework allows for the gradual incorporation of details into the simplified models, enabling faster trajectory generation while maintaining accuracy in deformable object manipulation.

To evaluate the proposed methods, we conduct extensive experiments across a range of tasks involving ropes, rectangular cloths, and complex cloths such as t-shirts. First, in Sec. V-B we assess two geometric model simplification approaches: line-fitting for 1-D linear objects and mesh simplification for 2-D surface objects. Next, in Sec. V-C we compare the performance of our goal-conditioned action space reduction methods for motion planning against several baselines, including the original action space, random action space, and grid-based action space. In Sec. IV-D, we then evaluate the effectiveness of the simplified dynamics models within the proposed planning framework, comparing them to the original dynamics model and grid-based models. Following this, in Sec. V-E, we compare our goal-informed models with a learned dynamics model. Finally, in Sec.VI we establish a real-world experimental system integrating perception, motion planning, and control subsystems, demonstrating the effectiveness of our proposed methods in performing cloth folding tasks.

This paper builds upon our previous work [23], where we introduce methods for simplifying the geometric model and reducing the action space for motion planning. Specifically, our contributions in Wang et al. [23] include:

- A general method of goal-conditioned geometric model simplification for 1-D linear deformable objects (e.g., ropes) that uses a piece-wise line fitted model, and for 2-D surface deformable objects (e.g., cloth) that uses a mesh simplified model (Sec. IV-B);
- An action space reduction scheme integrated into a manipulation planning pipeline based on the simplified geometric model of the deformable objects (Sec. IV-C);
- An extension of this approach to multi-step planning with intermediate goal states, where action space reduction and model simplification are performed on successive intermediate goals (Sec. V-A).

In this paper, we further extend our approach by incorpo-

rating simplified dynamics models and an iterative planning framework, and compare these methods against additional baselines. The contributions that are novel in this paper are as follows:

- A systematic approach for constructing a simplified dynamics model (Sec. IV-D), based on the original dynamics model and the simplified geometric model.
- An iterative model simplification & motion planning framework that generates manipulation plans faster, effectively balancing computational cost and trajectory optimality (Sec. IV-A).
- An extensive evaluation of the proposed model simplification and motion planning framework in simulation, covering a range of representative tasks involving ropes and cloth, using both search-based and optimization-based motion planning methods (Sec. V).
- A closed-loop experimental system incorporating perception, planning, and control for a Franka Panda robot performing cloth folding in the real world (Sec. VI).

II. Related work

While impressive progress has been achieved for rigid object manipulation, the development of deformable object manipulation methods lags behind, primarily due to two main challenges: high-dimensional state space and complex dynamics. These challenges make both planning and perception complex problems, particularly for traditional optimization or search based methods [24]–[26]. Comprehensive surveys of deformable object modeling, planning, control, and learning can be found in the works of Zhu et al. [15], Yin et al. [27], Bhagat et al. [28], and Arriola-Rios et al. [13].

The idea of identifying important points or features on deformable objects has been studied before. One way of approaching this problem is to determine particular features for a specific task. For example, Qiu et al. [20] and Sun et al. [29] propose to detect and eliminate wrinkles, where the task is to flatten a deformable object. Other approaches include using manually input key points [19], or contours [30]. Recently, learning-based methods for deformable object manipulation have been widely used [31]–[36]. Some of these approaches aim to simplify deformable object representations and identify key features on them. For example, Yan et al. [31] propose extracting a compact representation of the deformable object directly from raw sensor inputs for dynamics learning to facilitate faster planning. Lips et al. [32] learn key points from RGB images directly for specific cloth categories using synthetic data and manipulate deformable objects with scripted motion primitives. Zhou et al. [33] introduce a latent representation for soft object manipulation with semantic correlations, while Arnold et al. [34] estimate mesh representations from voxel inputs, and perform planning in mesh format internally. Ma et al. [35] approximate a deformable object as a sparse set of interacting key points and learn a graph neural network that captures the geometry abstractly. Additionally, Li et al. [36] propose learning 3D features using a combined PointNet encoder and neural radiance field (NeRF) for various deformable objects.

In this work, we also exploit the idea of identifying important features to reduce the action space. However, our approach differs from the aforementioned methods in that our method 1) can adapt to different tasks (as opposed to identifying features for one specific task); 2) identifies the key particles autonomously; and 3) takes a model-based (as opposed to learning-based) *and* goal-conditioned approach.

The idea of using coarse or approximate dynamics models in deformable object manipulation has gained significant attention, largely because high-fidelity models are scarce, difficult to fine-tune, and computationally expensive to simulate [37]. As a result, various simplified models have been proposed for specific tasks [21], [38]–[40]. Ruan et al. [21] develop a directional diminishing rigidity model for rope and cloth dragging without relying on simulating expensive mass-spring models. McConachie et al. [38] employ a virtual elastic band model to approximate true dynamics and train a classifier to decide when to trust the simplified model for rope manipulation in clutter. Additionally, McConachie et al. [39] introduce a multi-armed bandit method to adaptively select from multiple simplified models, avoiding the need for a high-fidelity deformable model in applications like rope winding and table covering. Power et al. [40] utilize simple models for cost-effective data collection, e.g. using a pendulum model to approximate a tethered rope, thereby enhancing learning efficiency. Zhou et al. [41] focus on identifying, modeling, and manipulating structures of interest rather than the entire object for bimanual bag manipulation, which substantially reduces computational load compared to full dynamics modeling. Learning-based methods have also been explored to train a neural network as the underlying dynamics model for motion planning [31], [36], [42]. For instance, Hoque et al. [42] demonstrate a visual dynamics model trained on domain-randomized RGBD images for fabric folding tasks, while Li et al. [36] use a recurrent state-space model (RSSM) for latent dynamics modeling. Other approaches involve using graph neural networks to learn deformable dynamics based on key points or mesh representations [35], [43]–[45]. Mitrano et al. [46] propose adapting a learned dynamics model to novel domains by focusing on the regions where source and dynamics are similar. Lee et al. [47] propose an efficient cloth simulation method using a miniature model with similar physical properties and an upscaling deep neural network. Their method focuses on simulating draped cloth scenarios, such as curtains and flags. Interests in model reduction extend to related areas in robotics, such as the framework proposed by Chen et al. [48], which aims to optimize reduced-order models and trajectories simultaneously for bipedal locomotion, and the application of model order reduction techniques to solve optimal control problems for cable-driven soft robots, significantly improving computational efficiency [49].

Instead of manually constructing simplified models or training neural network models offline for specific tasks, we base our dynamic model simplification on the various goals and tasks provided, allowing it to operate online without the need for offline training. Furthermore, our method can progressively refine the simplified model in a consistent way. These properties enable our method to accept arbitrary goals for the object

and to quickly perform model simplification conditioned on these goals, while gradually reducing the behavioral gap from the original dynamics models.

Recent advances in end-to-end vision-language-based systems have shown promise in robotic object manipulation [50], [51]. A particular strength of these approaches is that they do not require explicit dynamics modeling or state estimation, which is especially advantageous for deformable object manipulation. However, they still face challenges such as the need for large-scale annotated datasets, high computational demands, slow inference, and sensitivity to noisy sensory inputs. As a complementary approach, we simplify the dynamics and reduce the action space in a goal-conditioned manner, thereby avoiding the need for extensive data collection and model training.

III. Problem Formulation

We consider the motion planning problem of manipulating a deformable object into a given goal state. While the manipulation takes place in 3-D space, we focus on objects that can be represented by 1-D lines (like ropes) or 2-D surfaces (like cloths). For example, this could involve straightening a crumpled rope (Fig. 7(a)) or folding a piece of cloth diagonally into half (Fig. 7(c)).

In this section, we define the state space, action space, state transition function, and objective function of the manipulation problem. We distinguish between the *geometric model* of a deformable object and its *state*. The geometric model represents the connections between the particles in the mass-spring model and is topological, therefore is not affected by manipulation. The state of the model, on the other hand, represents the position of all particles, and is affected by manipulation actions.

For both 1-D linear and 2-D surface deformable objects, the geometric model can be represented as an undirected and weighted graph $\mathcal{G} = (V, E, L)$, where $V = \{1, 2, ..., N\}$ denotes a set of particles indexed from 1 to $N = |V|$, $E \subseteq \{\langle i, j\rangle \,|\, i, j \in V \; and \; i \neq j\}$ denotes the edges, and $L \in \mathbb{R}^{|E|}$ represents the edge weights, corresponding to the resting length of the edges.

The state of the deformable object at each time step during manipulation is defined by the positions of all particles, denoted as $\xi^t := \{p_i \mid \forall i \in V\}$, where $p_i = (x_i, y_i, z_i)$ represents the position of the i^{th} particle.

As for the action space $\mathcal{A}$, we assume the robot is equipped with a single gripper, which can pick any given particle in the object, that is, any $i \in V$, and move it by a certain distance along a direction in 3D space. We also add a 'None' action, corresponding to not holding any particle. The action at time t can be represented as:

$$a^t = \begin{cases} \langle i, \delta x, \delta y, \delta z\rangle \\ \text{None} \end{cases} \tag{1}$$

It is worth noting that the action space can be easily extended accommodate multiple grippers. Moreover, we constrain the movement of the gripper at each step to avoid moving the deformable object drastically: $|\delta x| \leq \Delta x$, $|\delta y| \leq \Delta y$, $|\delta z| \leq \Delta z$; Δx, Δy and Δz are motion limits along each axis in Cartesian space.

The dynamics model of deformable objects considered in this paper is based on a mass-spring system, which determines how the object moves and deforms in response to actions. We define it as $\mathcal{D} = (\mathcal{G}, \mathcal{M}, \mathcal{R})$, of which $\mathcal{G}$ denotes the geometric model as described above, $\mathcal{M}$ denotes the dynamics properties of all particles such as mass, and $\mathcal{R}$ denotes various dynamics relations between different particles such as the springs and collision constraints.

Based on the dynamics model $\mathcal{D}$, a state transition function can be defined accordingly:

$$\xi^{t+1} = f_{\mathcal{D}}(\xi^t, a^t). \tag{2}$$

which outputs the new state of the object given the current state ξ^t and action a^t.

We formulate the manipulation problem as a finite-horizon motion planning problem, whose solution is a trajectory, i.e., a sequence of T actions, $\tau = \langle a^0, a^1, ..., a^{T-1}\rangle$, that minimizes the distance between the final state of the particles ξ^T after executing τ and their goal state ξ^G:

$$\begin{aligned} \min_{\tau} \quad & \|\xi^G - \xi^T\| \\ s.t. \quad & \xi^{t+1} = f_{\mathcal{D}}(\xi^t, a^t), \; \forall t \in [0, T-1]\,, \end{aligned} \tag{3}$$

where we assume ξ^0 is given as the initial state, and the state transition function $f_{\mathcal{D}}$ guarantees the feasibility of states along the trajectory.

In this planning framework, the size of the action space is $|\mathcal{A}| = (N \times \mathbb{R}^3 + 1)$. The number of pickable particles affects the size of the action space linearly, which has, in turn, an exponential effect on the search space through the branching factor. Therefore, the number of picking points considered for planning has a significant impact on planning efficiency. Furthermore, the complexity of the dynamics model used in Eq. 3 also affects the computational cost greatly, as the motion planner requires massive amounts of trajectory rollouts.

In the next section, we present a methodology to decrease the computational cost of motion planning by reducing the action space and simplifying the dynamics model.

IV. Methodology

To alleviate the computational complexity of the motion planning problem for deformable object manipulation, as defined in Eq. 3, we employ two primary strategies: reducing the action search space $\mathcal{A}$ and utilizing faster, simplified dynamics models $\mathcal{D}$.

Our approach makes these reductions in an informed manner by leveraging goal-specific information to select key picking points (thereby reducing the action search space) and to simplify the dynamics model accordingly. We propose an iterative framework for model simplification and motion planning that progressively adds detail to the simplified model until a satisfactory trajectory is identified, ensuring both efficiency and effectiveness in solving the motion planning problem for deformable object manipulation.

In Sec. IV-A, we outline the overall iterative approach, while the subsequent sections detail its core components: Sec. IV-B

introduces the geometric model simplification pipeline for 1-D and 2-D objects, which serves as the foundation for both action space reduction (Sec. IV-C) and dynamics model simplification (Sec. IV-D); Sec. IV-C presents the action space reduction process; Sec. IV-D explains how the simplified dynamics model is constructed; lastly, Sec. IV-E describes the process of refining and combining the simplified models.

A. Iterative Model Simplification & Motion Planning

The overall process of the proposed action space reduction and iterative dynamics model simplification for motion planning is illustrated in Alg. 1.

This framework starts by computing the simplified geometric model $\hat{\mathcal{G}}_S$ based on the goal state of the original model (Line 2), and then uses this simplified geometric model to reduce the set of picking points $V_O^{\mathcal{A}}$ on the original model (Line 3). Next, a simplified dynamics model $\mathcal{D}_S$ is constructed using the simplified geometric model $\hat{\mathcal{G}}_S$ and the original dynamics model $\mathcal{D}_O$ (Line 5). The initial state ξ_O^0, the goal state ξ_O^G, and the reduced picking points $V_O^{\mathcal{A}}$ are then mapped to the simplified model based on the correspondence between the two models (Line 6). A planner is subsequently invoked to generate a trajectory that moves the simplified model from the initial state toward the goal within the reduced action space (Line 7). The planned trajectory is then rolled out on the original dynamics model (Line 8). During each loop, the original dynamics model, which is computationally expensive to simulate, is called only once, while numerous trajectory rollouts within the planner use coarse but computationally cheap models. If a satisfying trajectory is found or marginal improvement is observed for the original dynamics model, the process terminates (Line 10); otherwise, the geometric model simplification is invoked again to extract more details from the actually achieved final state (Line 11). The new simplified geometric model $\hat{\mathcal{G}}_S'$ is then combined with the previously simplified model $\hat{\mathcal{G}}_S$ to ensure that no information is lost from the previous step (Line 12). The trajectory planned from the previous simplified model is used as a warm start for the new iteration of motion planning, based on the updated simplified model (Line 7).

Algorithm 1: Iterative Model Simplification & Motion Planning

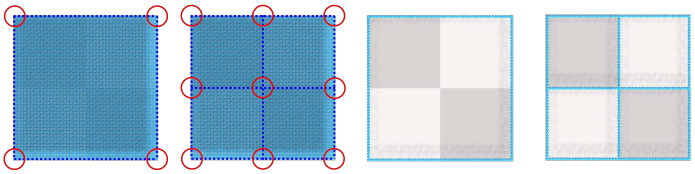
Input: $\mathcal{G}_O$, $\mathcal{D}_O$, ξ_O^0, ξ_O^G
Output: τ

1 $\tau \leftarrow None$;
2 $\hat{\mathcal{G}}_S, \hat{\xi}_S \leftarrow$ `Simplify_Geometry`$(\mathcal{G}_O, \xi_O^G)$;
3 $V_O^{\mathcal{A}} \leftarrow$ `Reduce_Action_Space`$(\mathcal{G}_O, \xi_O, \hat{\mathcal{G}}_S, \hat{\xi}_S)$;
4 **do**
5 $\quad \mathcal{D}_S \leftarrow$ `Simplify_Dynamics`$(\mathcal{D}_O, \hat{\mathcal{G}}_S)$;
6 $\quad \xi_S^0, \xi_S^G, V_S^{\mathcal{A}} \leftarrow$ `Map`$(\mathcal{D}_O, \mathcal{D}_S, \xi_O^0, \xi_O^G, V_O^{\mathcal{A}})$;
7 $\quad \tau \leftarrow$ `Planner`$(\mathcal{D}_S, V_S^{\mathcal{A}}, \xi_S^0, \xi_S^G, \tau)$;
8 $\quad \xi_O \leftarrow$ `Rollout`$(\mathcal{D}_O, V_O^{\mathcal{A}}, \tau)$;
9 $\quad$ **if** ξ_O^G *reached* `or` *convergent* **then**
10 $\quad\quad$ *break*;
11 $\quad \hat{\mathcal{G}}_S', \hat{\xi}_S' \leftarrow$ `Simplify_Geometry`$(\mathcal{G}_O, \xi_O)$;
12 $\quad \hat{\mathcal{G}}_S \leftarrow$ `Combine_Geometry`$(\hat{\mathcal{G}}_S', \hat{\mathcal{G}}_S)$;
13 **while** *True*;

An example of this iterative framework is shown in Fig. 1 for the task of folding a piece of cloth in half sideways. In the Geometric Model Simplification block, the original model at the folded goal state, which consists of a large number of particles, is reduced to a simplified geometric model with six triangles and eight particles. The corresponding particles on the original model are then extracted as potential picking points in the reduced action space, as shown in the Reduced Action Space block. A simplified dynamics model is then constructed based on the simplified geometric model, with particles and springs placed along the edges (first image in the Simplified Dynamics Model block of Fig. 1). Furthermore, the reduced picking points from the original model are mapped onto the simplified dynamics model, marked by red circles. Using the reduced action space and simplified dynamics model, a planner is invoked to generate a trajectory, which is then rolled out on the original model. As is shown in the Final State 1 of Fig. 1, the initially planned trajectory does not bring the cloth to the goal state. Thus, a new simplified model is created by incorporating more details from the actually achieved final state. A new trajectory is then planned based on this updated simplified dynamics model, which improves a bit, but still not satisfying. The same model simplification and planning loop runs another iteration, further refining the model and trajectory, ultimately bringing the original model closer to the goal and making it ready for execution on the real robot.

While our main methodology is goal-informed, as described above and illustrated in Fig. 1, the framework in Alg. 1 is general and does not depend on specific algorithms for action space reduction, dynamics model simplification, or the underlying planning methods. A baseline implementation involves uniformly reducing the action search space and simplifying the granularity of the dynamics model. For instance, when manipulating a piece of cloth, this could mean limiting the picking points to the particles on a coarse grid. As shown in Fig. 2(a-b), grids of different resolutions are overlaid on a piece of square cloth, with the intersection particles (denoted by the red circles) serving as potential picking points in the action space. Similarly, the original dense dynamics model could be replaced by grid-based models with coarse dynamics. As illustrated in Fig. 2(c-d), a set of grid-based dynamics models is constructed, featuring fewer particles and hollow interiors. Among these reduced models, a lower-resolution grid offers faster computation but lower accuracy, while a higher-resolution grid provides better precision at the cost of increased computation time.

However, this uniform reduction ignores the task/goal: certain action opportunities and certain details of the dynamics that are important for reaching the goal might be lost. Instead, our goal-conditioned implementation makes full use of available information from the goal or final state to improve



(a) Grid action 2×2 (b) Grid action 3×3 (c) Grid model 2×2 (d) Grid model 3×3

Fig. 2. Grid-based action space reduction & model simplification. In (a) and (b), a grid of varying resolutions is overlaid onto the underlying cloth model (shown in light blue), which consists of thousands of particles. The particles intersected by the grid on the original model are extracted as potential picking points, marked by red circles. In (c) and (d), grid-based dynamics models of different resolutions are shown, featuring particles aligned along the edges in light blue color with hollow spaces internally. This simplification maintains a general approximation uninformed by specific goals and tasks.

efficiency and accuracy.

In the following sections, we presents the details of our implementation, including the geometric model simplification function (`Simplify_Geometry`), the action space reduction function (`Reduce_Action_Space`), and the dynamics model simplification function (`Simplify_Dynamics`).

B. Geometric Model Simplification

The aim of simplifying the geometric model is to create an approximation of the original model while maintaining simplicity. A grid model of a certain resolution can only approximate certain shapes; for example, a grid geometry of resolution 2×2 cannot accurately represent the state of a side-folded cloth, as shown in Fig. 2(a). We adopt a goal-conditioned approach to simplify the geometric model so that it aligns well with the original model at the goal state.

The basic process of our informed geometric model simplification method, `Simplify_Geometry`, is illustrated in Alg. 2, where the inputs include the original geometric model $\mathcal{G}_O$ and the given state of the original model ξ_O, which can either be the goal state or the actually achieved final state, and the outputs are the simplified geometric model $\hat{\mathcal{G}}_S$ and the corresponding state of the simplified geometric model $\hat{\xi}_S$ Firstly, the algorithm tries to reduce the original geometric model to its simplest form, in which the number of target simplification elements is 2 (Line 1). A simplification function (`Simplify`) is then called to reduce the original geometric model $\mathcal{G}_O$ based on the given state ξ_O and the target number of elements N_S (Line 3). The distance between the given state of the original geometric model and the simplified model is calculated as the $error$, indicating how good a fit the simplified geometric model is to the original model, at the given state (Line 4). Lastly, if the error falls below a user-defined threshold or if the approximation fails to improve for a certain number of iterations—indicating convergence—the simplification process terminates (Line 6); otherwise, the process repeats with a more complex simplified geometric model (Line 5) by incrementing N_S. Overall, the `Simplify_Geometry` method adaptively finds a value of N_S that is sufficient to represent the given state ξ_O.

The method we use to simplify the geometric model differs for objects that can be approximated with 1-D models and objects that can be approximated with 2-D models, as detailed below.

Algorithm 2: Geometric Model Simplification (`Simplify_Geometry`)

Input: $\mathcal{G}_O$, ξ_O
Output: $\hat{\mathcal{G}}_S$, $\hat{\xi}_S$

1 $N_S = 2$;
2 **do**
3 $\quad \hat{\mathcal{G}}_S, \hat{\xi}_S \leftarrow$ `Simplify`$(\mathcal{G}_O, \xi_O, N_S)$;
4 $\quad error \leftarrow$ `Error`$(\hat{\xi}_S, \xi_O)$;
5 $\quad N_S \leftarrow N_S + 1$;
6 **while** *error* $>$ *threshold and not convergent*;

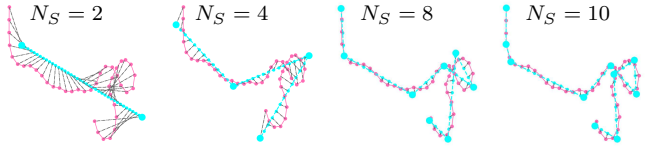


Fig. 3. Linear fitting of the simple model (blue) for a given goal state of the original model (pink), for different N_S values. Blue dots represent sampled points in the simplified model, and grey dashed lines connect the corresponding points of both models.

Piece-wise Line Fitting for 1-D Linear Models: For 1-D linear models, we adopt a piece-wise line fitting method to implement the `Simplify` function (Alg. 2, Line 3). Given the number of key particles N_S of the target simplification, a Quadratic Programming (QP) problem is defined and solved to find the optimal position of the N_S particles. The cost function of the QP problem to be minimized is the distance between the fitted piece-wise lines and the original shape of the object. To find such a distance value, we sample a number of, N_E, points on the two models. For example in Fig. 3, the pink dots show the N_E points sampled on the original model, while the blue dots show the N_E points sampled on the simplified model. We find the mean distance between corresponding particles (grey dot-dash line in Fig. 3):

$$distance(\hat{\xi}_S, \xi_O) = \sum_{i=1}^{N_E} \frac{\|\hat{q}_S^i - q_O^i\|}{N_E}, \tag{4}$$

where q_O^i represents the position of the i^{th} sampled point on the original model, and $\hat{q}_S^i$ represents the position of the i^{th} point sampled on the simplified model.

After the QP minimization is complete, we use the same distance formulation above (Eq. 4) to implement the `Error` function in Alg. 2 (Line 4) to compute the final distance between the two models.

An example piece-wise line fitting process for a rope of random shape is shown in Fig. 3. The first picture shows a simplified model with two particles and one line segment; the fit to the original model is quite poor. As the number of particles in the simplified model, N_S, is increased, fits improve. Depending on the threshold set (Alg. 2, Line 6), the simplification process can terminate, for example, at the ten-particle model ($N_S = 10$) in Fig. 3.

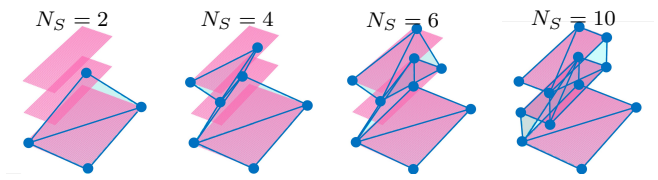


Fig. 4. Reduced mesh of the simple model (blue) for a reflectively folded state of the original model (pink), for different N_S values. The vertical dimension in these figures are scaled up for visualization purposes, to make the three layers of the folded cloth visually separate.

Mesh Simplification for 2-D surface Models: For objects that can be approximated by surfaces, we use the Quadric Edge Collapse Decimation (QECD) method as the geometric model simplification function, `Simplify`, of Alg. 2, which can simplify the model towards a given number of elements N_S. The basic element for mesh simplification is a triangle, which is composed of particles that can be shared between different triangles. QECD is a surface simplification algorithm based on the quadric error metrics proposed by Garland and Heckbert [52]. During simplification, pairs of vertices (particles) are contracted to one iteratively, until the target number of triangles, N_S, is achieved. We use the QECD implementation in the mesh processing library Meshlab [53].

We start from simplifying the original mesh to a model with $N_S = 2$ triangles (four particles, two of which are shared), and gradually increase the number of triangles until the error is below the threshold, as in Alg. 2.

To implement the `Error` function and find the distance between the simplified mesh and the original one (Alg. 2, Line 4), we use the Hausdorff distance [54], which is a widely used similarity metric for mesh and image comparison. The Hausdorff distance is defined as the maximum distance of a set to the nearest point in the other set, and in our case, is found by:

$$h(\hat{\xi}_S, \xi_O) = \max_{\hat{q}_S \in \hat{\xi}_S} \{ \min_{q_O \in \xi_O} \|\hat{q}_S - q_O\| \}. \tag{5}$$

As shown in Fig. 4, a piece of reflectively-folded cloth (i.e., when the cloth is first folded in half, and then the top half is folded a quarter back onto itself) is initially approximated by a simple mesh with two triangles, which only covers the bottom face. This is improved by adding more triangles to the simplified model. In the last picture, a simplified model with ten triangles overlaps with the original model, giving us a satisfying approximation.

C. Action Space Reduction

The action space for manipulating a deformable object, as defined in Eq. 1, consists of a picking index and a 3-D vector in Cartesian space. Given the potentially high number of pickable points on a deformable object, this action space can be very large. Thus, we propose the `Reduce_Action_Space` function to extract specific particles from the deformable object as the potential picking points based on the simplified geometric model at the goal state. In this way, the particles that are relevant to achieving the goal are automatically selected as picking points in the action space.

To extract these key particles, we first align the reduced geometric model with the original geometric model at the goal state and calculate the distances between the vertices of both models to identify the key particles on the original model:

$$V_O^{\mathcal{A}} \leftarrow \bigcup_{\hat{v}_s \in \hat{V}_S} \arg\min_{v_o \in V_O} \|\hat{\xi}_S(\hat{v}_s) - \xi_O(v_o)\| \,, \tag{6}$$

where $\hat{V}_S$ denotes the vertices of the simplified geometric model $\hat{\mathcal{G}}_S(\hat{V}_S, \hat{E}_S, \hat{L}_S)$; V_O denotes the vertices of the original geometric model $\mathcal{G}_O(V_O, E_O, L_O)$; $\hat{\xi}_S$ denotes the state of the simplified geometric model, and $\hat{\xi}_S(\hat{v}_s)$ denotes the position of vertex $\hat{v}_s$; ξ_O denotes the state of the original geometric model, and $\xi_O(v_o)$ denotes the position of vertex v_o.

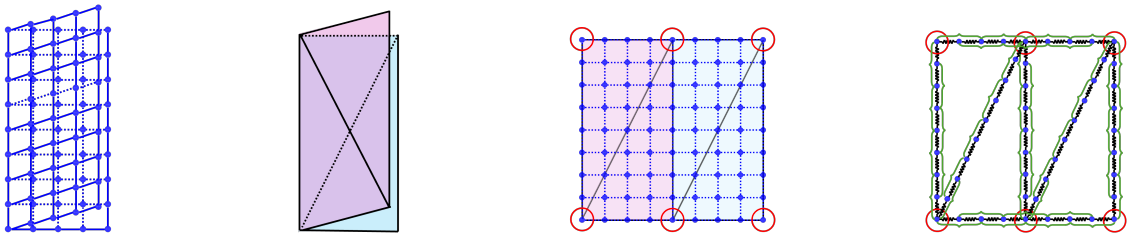
We then use $V_O^{\mathcal{A}}$ as the picking points in the reduced action space $\mathcal{A}_S$, which are subsequently used for motion planning. Compared to using all particles in the original model as picking points in the action space, the search space is significantly reduced by confining the picking points to the reduced key particles, where $|V_O^{\mathcal{A}}| \ll |V_O|$. We only invoke the action space reduction function once for the original geometric model, as the key particles can then be correspondingly extracted for other models.

An example of this process is illustrated in Fig. 5, where the original geometric model at the goal state is illustrated in Fig. 5(a), while a simplified model with four triangles, obtained using Alg. 2, is depicted in Fig. 5(b). Six particles on the original geometric model are then identified which are marked by red circles in Fig. 5(c). These particles are similarly identified in another simplified model (generated in Sec. IV-D), demonstrating that this approach maintains a consistent action space across different but related models.

D. Dynamics Model Simplification

Before we explain how the `Simplify_Dynamics` function in Alg. 1 works, we will first explain how the original dynamics model $\mathcal{D}_O(\mathcal{G}_O, \mathcal{M}_O, \mathcal{R}_O)$ and the original geometric model $\mathcal{G}_O$ are related. As illustrated in Fig. 6, for each vertex of the geometric model, a particle with mass and radius is generated in the dynamics model. All particles in the dynamics model are assigned the same mass depending on the materials and properties of the object (giving us the dynamics properties of particles $\mathcal{M}_O$). Then, for neighboring particles, a stretching spring is added; for particles that are two steps away, a bending spring is added; for diagonal neighboring particles, a shearing spring is added; additional constraints of self-collision and environmental collision are also considered (giving us the dynamics relations of particles $\mathcal{R}_O$).

We build the simplified dynamics model $\mathcal{D}_S$ similarly, as detailed in Alg. 3, given the reduced geometric model $\hat{\mathcal{G}}_S$ and the original dynamics model $\mathcal{D}_O$. However, the reduced geometric model $\hat{\mathcal{G}}_S$ can include edges that are significantly long; e.g., the nine edges in Fig. 5(b). To express the deformation capability of these edges in the dynamics model, we construct a model with interpolated particles along these edges (blue points in Fig. 5(d)). Thus, given each edge $\hat{e}(u, v)$



(a) Original geometric model (b) Simplified geometric model (c) Reduced action space (d) Simplified dynamics model

Fig. 5. Model simplification for cloth side folding. The original geometric model at the folded state is shown in (a); we simplify it using the proposed Alg. 2 and get a simplified geometric model in (b); by extracting the nearest particle on the original model, six particles on the original model are identified as picking points in the reduced action space, as shown in (c); a simplified dynamics model is built based on the simplified geometric model, with particles and springs along the edges, and the corresponding particles for picking are extracted accordingly, as illustrated in (d).

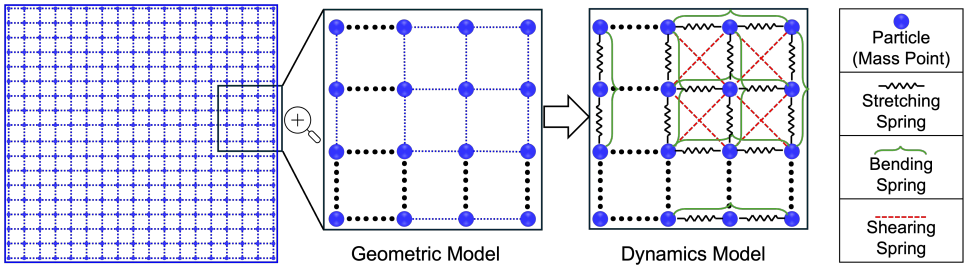


Fig. 6. Original geometric model and dynamics model. The geometric model is composed of particles and edges with weights assigned to each edge representing the resting length, while the dynamics model builds on this structure by assigning mass to each particle and establishing various spring connections between these particles.

Algorithm 3: Dynamics Model Simplification (`Simplify_Dynamics`)

Input: $\mathcal{D}_O(\mathcal{G}_O, \mathcal{M}_O, \mathcal{R}_O)$, $\hat{\mathcal{G}}_S(\hat{V}_S, \hat{E}_S, \hat{L}_S)$
Output: $\mathcal{D}_S(\mathcal{G}_S, \mathcal{M}_S, \mathcal{R}_S)$

1 $V_S \leftarrow \hat{V}_S$; $E_S, L_S, \mathcal{M}_S, \mathcal{R}_S \leftarrow None$;
2 **for** $\hat{e}(u,v)$ *in* $\hat{E}_S$ **do**
3 $V_S^{\hat{e}} \leftarrow None$; u_{new}, $v_{new} \leftarrow u$;
4 **do**
5 $w_{new} \leftarrow u_{new}$, $u_{new} \leftarrow v_{new}$;
6 $v_{new} \leftarrow$ `New_Vertice`$(u_{new}, radius)$;
7 $V_S \leftarrow V_S \cup \{v_{new}\}$, $V_S^{\hat{e}} \leftarrow V_S^{\hat{e}} \cup \{v_{new}\}$;
8 $E_S \leftarrow E_S \cup \{(u_{new}, v_{new})\}$;
9 $L_S \leftarrow L_S \cup \{radius\}$;
10 $\mathcal{R}_S \leftarrow \mathcal{R}_S \cup$ `Add_Strech`$(u_{new}, v_{new}) \cup$ `Add_Bend`(w_{new}, v_{new});
11 **while** $\hat{L}_S(u,v) - L_S(u, v_{new}) > radius$;
12 $\mathcal{M}_S^{\hat{e}} \leftarrow$ `Mass_Aggregate`$(\mathcal{M}_O, \hat{e})$;
13 $\mathcal{M}_S \leftarrow \mathcal{M}_S \cup$ `Mass_Average`$(\mathcal{M}_S^{\hat{e}}, V_S^{\hat{e}})$;
14 $\mathcal{G}_S \leftarrow (V_S, E_S, L_S)$, $\mathcal{D}_S \leftarrow (\mathcal{G}_S, \mathcal{M}_S, \mathcal{R}_S)$;

(Line 2) from the reduced geometric model $\hat{\mathcal{G}}_S$, we gradually add vertices along the edge from the starting vertex u to the end vertex v given a preset $radius$ (Line 5-6). This process results in a more detailed set of vertices V_S, as well as corresponding edges E_S and rest lengths L_S (Line 7-9). Each newly added vertex corresponds to a particle, to which we apply stretching constraints (black springs in Fig. 5(d)) for neighboring particles and bending constraints (green connections in Fig. 5(d)) for particles two steps apart (Line 10). Next, we align the simplified geometric model with the original dynamics model at the flattened state, as shown in Fig. 5(c), and aggregate the masses of the particles from the original dynamics model, $\mathcal{D}_O$, that are closest to the current edge $\hat{e}(u,v)$ (Line 12):

$$V_O^{\hat{e}} \leftarrow \{v_o \in V_O | \arg\min_{\hat{e}' \in \hat{E}_S} dis(v_o, \hat{e}') = \hat{e}\} \tag{7}$$

$$\mathcal{M}_S^{\hat{e}} \leftarrow \sum_{v_o \in V_O^{\hat{e}}} \mathcal{M}_O^{v_o} \tag{8}$$

where $V_O^{\hat{e}}$ denote the vertices from the original model that are closest to edge $\hat{e}$, $M_O^{v_o}$ denotes the mass of v_o, and $\mathcal{M}_S^{\hat{e}}$ denotes the mass of edge $\hat{e}$. Then we distribute this mass equally among the particles added to the current edge (Line 13):

$$\mathcal{M}_S^{v} \leftarrow \frac{\mathcal{M}_S^{\hat{e}}}{|V_S^{\hat{e}}|}, \text{ for } v \in V_S^{\hat{e}}, \tag{9}$$

where $V_S^{\hat{e}}$ denotes the set of vertices expanded along the edge $\hat{e}$ of the simplified geometric model. We avoid redistributing mass for each particle separately based on the distance between particles in the original and simplified models, as this can lead to uneven mass distribution and degrade the performance of the simplified dynamics model. At this point, we have the complete simplified geometric model $\mathcal{G}_S$, the mass distribution $\mathcal{M}_S$, and the dynamics relations $\mathcal{R}_S$, which give us the simplified dynamics model $\mathcal{D}_S$ for the target task (Line 15). As shown in Fig. 5(d) a simplified mass-spring model with only the particles along the edges is built, which, compared to the original model, has a much smaller number of particles and constraints, thereby allowing for faster simulation.

Algorithm 4: Geometric Model Combination (`Combine_Geometry`)

Input: $\hat{\mathcal{G}}'_S(\hat{V}'_S, \hat{E}'_S, \hat{L}'_S)$, $\hat{\mathcal{G}}_S(\hat{V}_S, \hat{E}_S, \hat{L}_S)$
Output: Updated $\hat{\mathcal{G}}_S$

1 **for** $v'_S \in \hat{V}'_S$ **do**
2 **if** $v'_S \notin \hat{V}_S$ **then**
3 $\hat{V}_S \leftarrow \hat{V}_S \cup \{v'_S\}$;
4 **for** $e'_S(u, v) \in \hat{E}'_S$ **do**
5 **if** $u \in \hat{V}_S$ && $v \in \hat{V}_S$ **then**
6 $\hat{E}_S \leftarrow \hat{E}_S \cup \{e'_S\}$;
7 **for** $(u, v) \in \hat{V}_S \times \hat{V}_S$ **do**
8 **if** $(u, v) \notin \hat{E}_S$ && *not Intersect*$(u, v, \hat{E}_S)$ **then**
9 $\hat{E}_S \leftarrow \hat{E}_S \cup \{(u, v)\}$;

E. Model Refinement and Combination

A key component of the iterative model simplification framework is model refinement and combination. When the planned trajectory leads the original model to a final state that is not sufficiently close to the goal, we reapply the geometric model simplification process, this time using the actually achieved final state as input (line 11 in Alg. 1).

To ensure consistency in the simplified model throughout the planning process, we retain the initially simplified geometric model while integrating additional details from the newly simplified model, as shown on line 12 (`Combine_Geometry`) in Alg. 1. The process follows three main steps, which correspond to the key operations in Algorithm 4 explained below.

Particle Integration (Lines 1–3): We first identify and incorporate new particles from the newly simplified geometric model, $\hat{\mathcal{G}}'_S$, that do not exist in the initially simplified model, $\hat{\mathcal{G}}_S$. This ensures that the refined model includes all relevant structural elements while retaining the original simplified representation.

Edge Retention (Lines 4-6): Once the particle set is updated, we retain edges from $\hat{\mathcal{G}}'_S$ where both end particles exist in the combined particle set. This step ensures that existing connections between incorporated particles are preserved, maintaining structural consistency.

Edge Completion and Consistency Check (Lines 7-9): After merging edges from both models, the new graph may contain incomplete triangular structures where necessary edges are missing. To address this, we add missing edges while ensuring that newly introduced edges do not intersect with existing ones. This step preserves the geometric integrity of the simplified model.

Through these steps, the refined geometric model retains all information from the initial model while incorporating additional structural details from the newly simplified version. This enhances the accuracy of subsequent planning iterations by better representing the actually achieved state.

V. Simulation Experiments

To validate the effectiveness of the proposed framework, extensive experiments and ablation studies with various baselines are conducted in simulation on a range of representative deformable object manipulation tasks for ropes and cloths. All simulation experiments are performed in SoftGym [49] on a workstation equipped with Intel(R) Core(TM) i9-11900 CPU @2.50GHz, 32 GB of RAM, and Nvidia GeForce GTX 4090 GPU.

In Sec. V-A, we briefly introduce the different tasks; in Sec. V-B, we present the results of the proposed geometric model simplification methods; in Sec. V-C we compare the performance of planning with our goal-conditioned action space against the uninformed baseline action spaces; in Sec. V-D, we evaluate the effectiveness of the entire framework with our proposed model simplification methods and the baseline methods; in Sec. V-E, we compare our simplified models with learned dynamics models.

A. Tasks

Seven representative tasks are considered, comprising two for ropes, four for square cloths, and one for a single-layer t-shirt. Among these tasks, Rope Straightening and Cloth Side Folding are borrowed directly from SoftGym, while the remaining tasks are created based on SoftGym.

1) Rope Straightening: Manipulating a rope from a randomized initial state to a straightened goal state (Fig. 7(a)). Performance (i.e., cost) is measured by the error between the distance of the two endpoints and the original length of the rope.

2) Rope Folding: Manipulating a rope into a crossed triangle (Fig. 7(b)), from an initially straightened state. Performance (i.e., cost) is measured based on the bipartite matching distance between the final and goal positions of all particles.

3) Cloth Diagonal Folding: Folding a flattened cloth into half diagonally (Fig. 7(c)). Performance (i.e., cost) is measured based on the distance between corresponding particles of the two triangular halves of the cloth, with an additional penalty for any drift of the bottom-side particles from the cloth's initial position, as similarly described in [55]. To compute the overall cost, we used a weight of 1.0 for the averaged distance between corresponding particles and a weight of 1.2 for the averaged drift penalty from the initial position. For all the other tasks below, we used these same weights.

4) Cloth Side Folding: Folding the flattened cloth on the table into half sideways (Fig. 7(d)). The cost function is similar to that of Cloth Diagonal Folding, comprising the distance between corresponding particles of the two rectangular halves of the cloth and a penalty for any displacement of the cloth from its initial position [55].

5) Cloth Reflective Folding: Reflective folding is borrowed from origami folding where two consecutive folds are in opposite directions [56]. This task follows the goal state of the previous task (i.e., the side folding goal), with the final goal being to fold a quarter of the cloth back, as shown in Fig. 7(e). The cost is evaluated based on the distance between corresponding particles across the three folded layers, along

with a penalty for dragging the cloth away from its initial position.

6) Cloth Underneath Folding: Underneath folding (e.g., Fig. 7(f)) refers to folding a quarter of the cloth under the rest of it, contrary to placing it on the top. This task builds upon the final state of the previous task (i.e., Cloth Reflective Folding) and showcases multi-step planning with intermediate goals. The cost function considers the distance between the underneath folded part and the ground, the alignment with corresponding particles on the top layer, and a penalty for any displacement from the initial pose.

7) Single Layer T-shirt Side Folding: This task involves manipulating a single-layer t-shirt, which has an irregular shape compared to the previous tasks. The goal is to fold the left-hand half of the flattened t-shirt on top of the right-hand half (Fig. 7(g)). The cost function follows the same structure as the Cloth Side Folding task.

In the above tasks, *Cloth Side Folding*, followed by *Cloth Reflective Folding*, followed by *Cloth Underneath Folding* can be interpreted as a sequence of sub-goals that combine into a long-horizon task. While these sub-goals are currently provided by us, the development of high-level sub-goal planners is a promising research direction.

B. Geometric Model Simplification

Geometric model simplification serves as the foundation for both action space reduction and dynamics model simplification. In this section, we aim to demonstrate quantitatively and qualitatively how good a fit the proposed geometric methods can give for various goals and tasks. Additionally, the time cost associated with these simplifications will be assessed to determine their significance relative to the time required for motion planning.

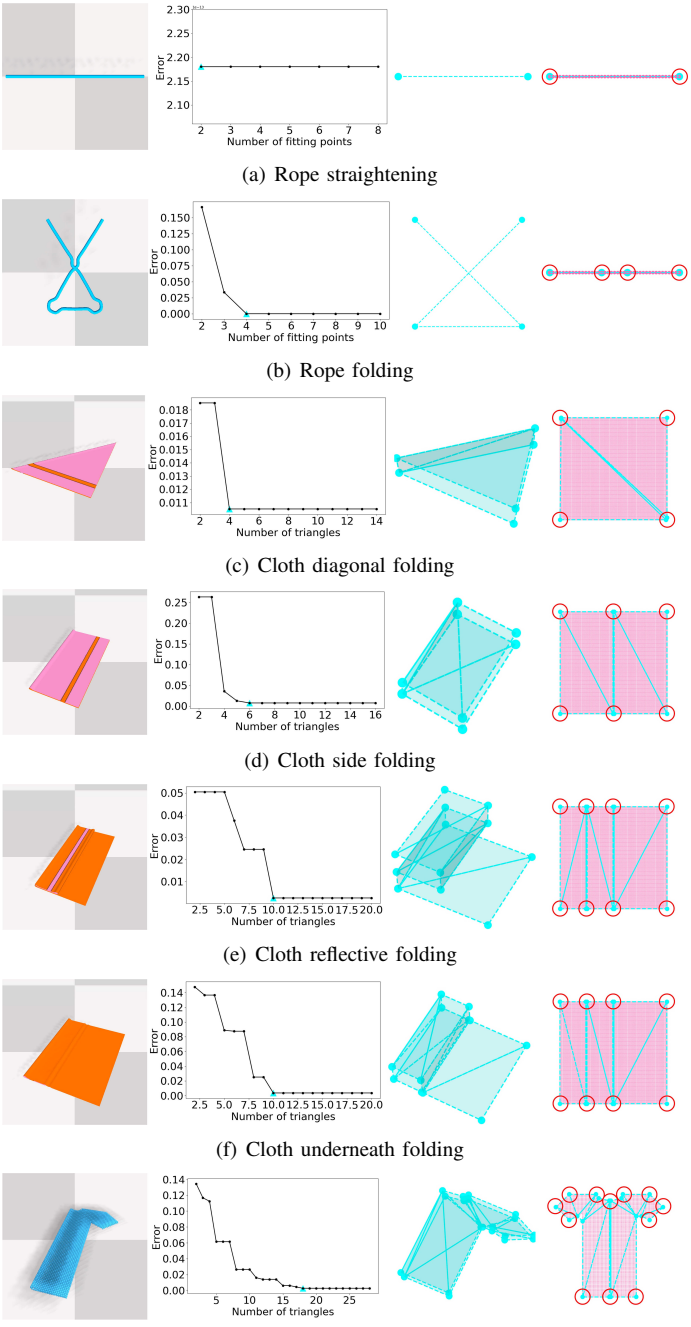
As the parameters of our method, we set the error threshold in Alg. 2 to a small value (0.001). Moreover, if the error in Alg. 2 does not improve (i.e., converges) for a certain number of steps (5 for 1-D linear objects, and 10 for 2-D surface objects), Alg. 2 also stops. These values can also be observed in the second column of Fig. 7.

The results of geometric model simplification given the goal states of different tasks are shown in Fig. 7. For the rope straightening task, as expected, only the two ends are enough to represent and reach the goal, as shown in Fig. 7(a). For the rope folding task, as depicted in Fig. 7(b), a simplified model of four particles and three line segments is acquired, identifying four corners on the original model as key particles.

As for cloth manipulation tasks, our method finds much simpler geometric models than the original model for each task, which is illustrated in Fig. 7(c-f). In Fig. 7(c), the Hausdorff distance curve flattens at four triangles, which provides a fine approximation of the original mesh model for the diagonal folding task. Corresponding key particles are extracted and marked with red circles in the figure. Similarly, for the rest of the cloth folding tasks, a set of simplified geometric models with six, ten, and ten triangles are acquired to approximate the original models respectively.

For the t-shirt which has a more complex shape than cloth, our method finds a simplified model consisting of eighteen triangles, which fits well to the original geometric model both at the goal state and the flat state.

The average geometric model simplification time costs for line-fitting and mesh simplification are 0.125 s and 0.730 s respectively, which are negligible compared to the time required for motion planning.



(a) Rope straightening

(b) Rope folding

(c) Cloth diagonal folding

(d) Cloth side folding

(e) Cloth reflective folding

(f) Cloth underneath folding

(g) T-shirt folding

Fig. 7. Geometric model simplification. The figure displays the goal images in SoftGym (left), the approximation error curve (middle left), the simplified geometric model at the goal state (middle right), and the simplified geometric model at the flattened state (right).

C. Action Space Reduction for Motion Planning

In this section, we conduct motion planning experiments to examine how the size and structure of the action space

influence planning efficiency and success across various tasks. We also compare the effectiveness of goal-informed and uninformed action space reduction methods to determine which approach yields better performance.

For a fair comparison across all action spaces and tasks, the experiments utilize the same underlying dynamics models and motion planners. Specifically, we employ the high-fidelity original dynamics model within the planner. And as our method is independent of the planners used, we implement two widely used algorithms, Cross-Entropy Method (CEM [57]) and Rapidly-exploring Random Tree (RRT [58]), of which the former is optimization-based and the latter is search-based.

Implementation & Baselines: For each planner, we compare our goal-informed action space with various baseline action spaces, as listed below.

- **Reduced Action Space (Ours)**: it refers to picking only the key particles, which are extracted by simplifying the geometric model at the goal state. For each task, the key particles are marked by red circles in the last column of Fig. 7.
- **Original Action Space**: this is the baseline where the gripper can pick any particle on the object.
- **Random Action Space**: in this baseline, we uniformly randomly sample the same number of picking points as the reduced action space, using it as a random comparison.
- **Grid Action Space**: instead of choosing picking points randomly, we overlay a grid of different resolutions on the original geometric model and select the particles nearest to the intersections as picking points. Three grid baselines are considered in this paper, including Grid Action Space (2×2), Grid Action Space (3×3), and Grid Action Space (4×4). Example picking points of grid action space are shown in Fig. 2(a-b) which are marked by red circles.

Furthermore, to demonstrate the effectiveness of the planned trajectory, we manually script a manipulation strategy based on the *Reduced Action Space* and denote it as **Scripted Policy**, in which the gripper picks each key particle, moves above its target position, and releases it.

Results: Since CEM and RRT are stochastic methods, the experiments are repeated ten times for each planner on each task. We first show the results using CEM, which we run for 30 iterations, with a population size of 400 for all experiments, and later discuss the results with RRT. The planned trajectory costs versus time for CEM with different action spaces are shown in Fig. 8 (a-g).

For rope straightening, *Reduced Action Space* finds a better plan than *Original Action Space* using much less time, which is shown in Fig. 8(a). The random action space does not converge to an acceptable solution; the *Original Action Space* converges but takes much longer time; the *Grid Action Spaces* converge at different speeds depending on the grid resolution. The *Scripted Policy* finds a slightly worse solution because it keeps disrupting previously achieved states.

For the rope folding task, *Reduced Action Space* still achieves a smaller cost. The *Original Action Space* converges to a similar cost as the *Grid Action Space (4×4)*, whereas the *Grid Action Space (2×2)* yields the worst solution, which reflects that a limited number of picking points are insufficient for handling complex folding tasks with ropes.

For the rope object, the size difference between the *Original Action Space*, ($N_O = 40$), and the *Reduced Action Space* ($\hat{N}_O = 2$ for the straightening task, and $\hat{N}_O = 4$ for the folding task), is not significant. Therefore we see a modest gain between the two methods. For the cloth object however, the original model has $N_O = 10000$ particles, which is much larger than the number of extracted key particles $\hat{N}_O$, as shown in Sec. V-B. Therefore, from Fig. 8(c-f), we can see that *Reduced Action Space* achieves a better solution than *Original Action Space* consistently across all four cloth manipulation tasks. For instance, as shown in Fig. 8(c), the planning cost of *Reduced Action Space* converges to an optimal plan in forty minutes, successfully folding the cloth to match the goal state. In contrast, *Original Action Space* converges to a plan with a much higher cost which results in an incorrect final state. As the difficulty increases from side folding to reflective folding and underneath folding tasks, the gap between the cost curve of *Reduced Action Space* and *Original Action Space* widens. Furthermore, in these more complex folding tasks, the advantage of our method over the simple scripted behavior is also increasingly evident.

In all cloth folding tasks, *Random Action Space* achieves slightly better results than *Original Action Space*, which further demonstrates that not every particle on the original model is equally relevant to achieving the goal. For different tasks, the resolution of the *Grid Action Space* has significantly different influences: for cloth diagonal and reflective folding, the *Grid Action Space (2×2)* provides a better solution; for side folding, *Grid Action Space (3×3)* is more effective; for underneath folding, *Grid Action Space (4×4)* achieves better results.

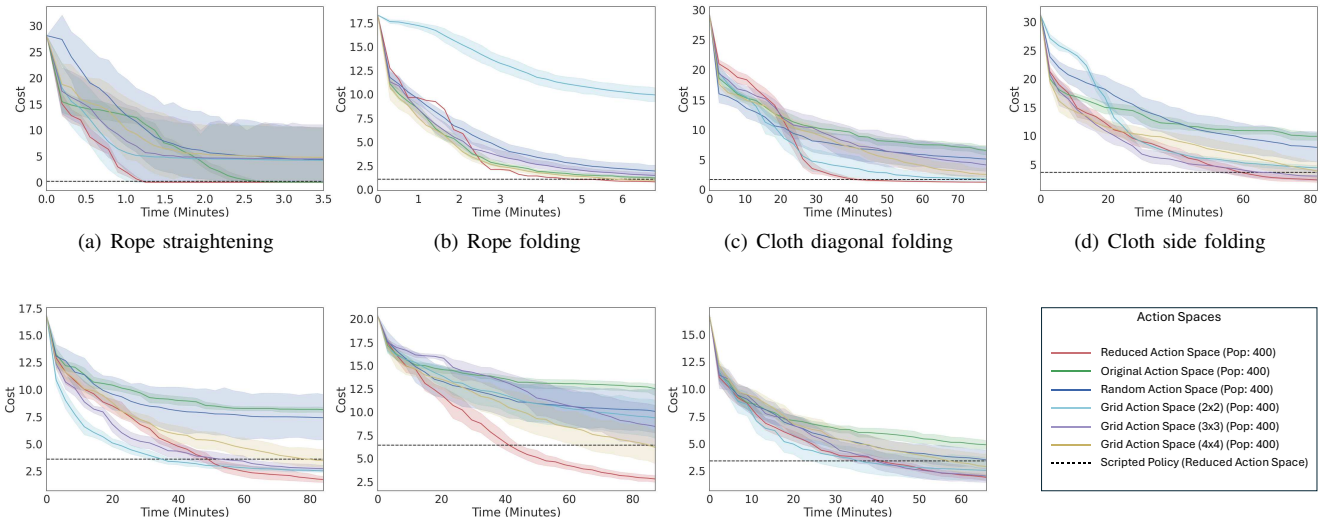
For t-shirt folding tasks, *Reduced Action Space* converges to a similar cost as the *Grid Action Space* of resolution 4×4, but faster. The *Original Action Space* results in the worst cost.

We also run experiments with RRT, a classic search-based planning method, to demonstrate that our method is independent of the underlying algorithms of the planner. We implement RRT from Open Motion Planning Library [59] (OMPL), and set the goal bias and error threshold to 0.5 and $1e^{-6}$ respectively. For each task and different action spaces, we run the planner from one minute to twenty minutes. The results of various action spaces for RRT are consistent with those from CEM-based planners, with *Reduced Action Space* consistently achieving the best cost across all tasks.

D. Iterative Model Simplification and Motion Planning

Based on the reduced action space, we conduct further experiments with the iterative model simplification and motion planning framework proposed in this paper.

The aims of these experiments include: 1) evaluate to what extent can the simplified dynamics models accelerate the motion planning compared to more complex, high-fidelity models; 2) the quality and effectiveness of the trajectories generated using simplified models; 3) compare the performance of different model simplification strategies—specifically, goal-



(a) Rope straightening (b) Rope folding (c) Cloth diagonal folding (d) Cloth side folding

(e) Cloth reflective folding (f) Cloth underneath folding (g) T-shirt side folding

Fig. 8. Planner (CEM) performance over time for different tasks using different action spaces.

informed versus uninformed dynamics model simplification—to identify which approach yields better plans for various tasks.

Implementation & Baselines: For the designed framework, we can change the dynamics model and motion planner flexibly depending on various requirements. Specifically, for dynamics models, we have the goal-informed simplified dynamics model, the original dynamics model, and the grid dynamics model of different resolutions and orientations. We conducted all experiments using both CEM and RRT. To ensure a fair comparison, all planners use the same action space, which is the `Reduced Action Space`. Details of our methods and baselines are listed below.

- **Simplified Model**: This is the implementation of the proposed framework, where the simplified dynamics model is iteratively improved and used for motion planning as in Alg. 1. To terminate, we use a small cost threshold (0.02) to determine whether the trajectory found is satisfactory. In addition to this threshold, the iterative method will also stop if it "converges" (i.e., the method stops if there are no improvements in the achieved cost for successive iterations (currently set to 5), even if the cost is not under the threshold).
- **Simplified & Original Model**: This is a variation of the proposed framework, where we only simplify the dynamics model once, use the planned trajectory based on the simplified model as a warm start, and continue optimizing it on the original dynamics model.
- **Original Model**: In this baseline, the original high-fidelity but time-consuming dynamics model is used for motion planning.
- **Grid Model**: For this baseline, instead of using simplified dynamics models informed by the goal, we use grid-based models, and gradually increase the granularity in the iterative planning framework. For example, a set of grid-based dynamics models of various resolutions is shown in Fig. 2(c-d). (We also implemented a triangular version of this method. However, since its performance was similar to the Grid Model, we do not include it in our results due to space constraints.)

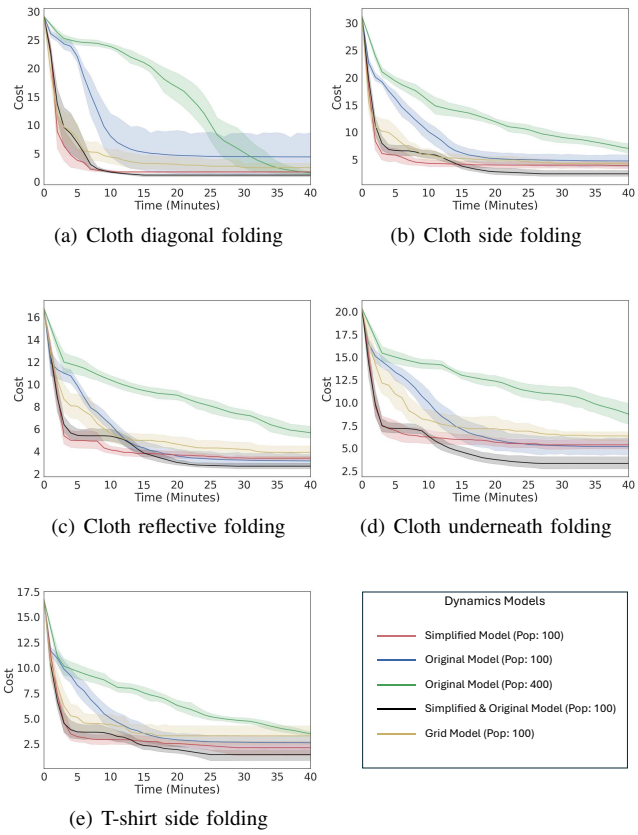
For model parameters such as mass and spring coefficients, we use the default values in SoftGym for both the original model and the reduced models (including our goal-informed simplified models and the grid simplified models). The default settings assign each particle a mass of 1.0, set the stretching coefficient to 0.8, the bending coefficient to 1.0, and the shear coefficient to 0.9.

Results: For all cloth folding and t-shirt folding tasks, we run the motion planning experiments with both CEM and RRT separately. We repeat the experiment using the proposed methods and the baseline methods for ten times.

The results of motion planning based on CEM are shown in Fig. 9. We reduce the population size of CEM to 100 to plan faster, while keeping a large population-sized *Original Model* baseline for better comparison.

From the results we can see that, for *Original Model*, CEM with a small population converges faster but results in a higher cost compared to CEM with a large population for all cloth folding and t-shirt folding tasks. *Simplified & Original Model* achieves the best cost for all tasks and converges more quickly than *Original Model*, demonstrating that using the simplified model as a warm start helps guide the planner to find better trajectories more efficiently on the original model. *Simplified Model* converges the fastest among all model selections for all tasks and achieves similar or better costs than *Original Model* with the same population size. *Grid Model* achieves acceptable cost for all tasks, however, it is not as efficient as the goal-conditioned simplified dynamics model, which demonstrates that simplifying the dynamics model in an informed way is more effective than universally simplified models.

The motion plans generated by our method are shown in Fig. 10 and Fig. 11 for cloth side folding and t-shirt side

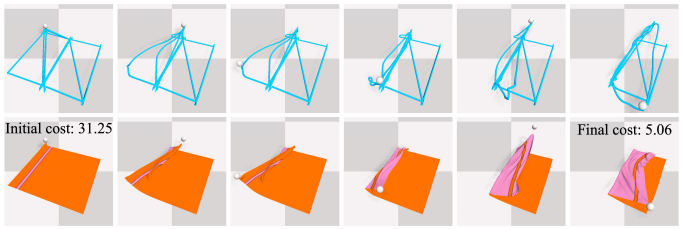


(a) Cloth diagonal folding (b) Cloth side folding

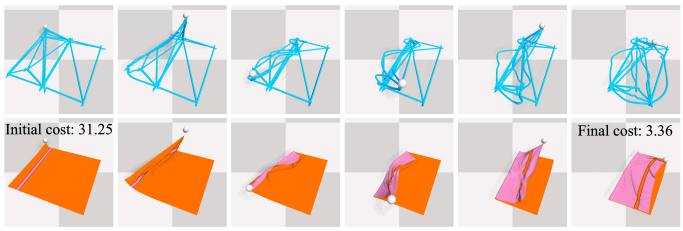
(c) Cloth reflective folding (d) Cloth underneath folding

(e) T-shirt side folding

Fig. 9. Planner (CEM) performance over time for different tasks using different dynamics models.



(a) First simplified model



(b) Second simplified model

Fig. 10. Cloth side folding with simplified models. In image (a), a trajectory is initially planned using the simplified model, depicted in the first row. This trajectory is then rolled out on the original model, shown in the second row, but it fails to achieve the desired goal. Consequently, a new round of model simplification is conducted based on the actual final state observed in the last image of the second row in (a). This results in a more detailed simplified model. Using this refined model, a new trajectory is successfully planned and executed, effectively bringing the cloth to the goal, as illustrated in image (b)

TABLE I
COMPARISON OF SINGLE SIMPLIFIED MODEL AND ITERATIVE SIMPLIFIED MODELS. RESULTS SHOW: FINAL TRAJECTORY COST ACHIEVED (NUMBER OF MODEL ITERATIONS REQUIRED).

Task	Diag.	Side	Reflec.	Undern.	T-shirt
Single	6.24	5.9	5.0	7.5	3.9
Iterative	1.76 (2.3)	3.8 (3.6)	3.8 (3.1)	5.3 (4.1)	2.5 (3.4)

folding respectively. Take the t-shirt folding for an example, as shown in Fig. 11(a), the first trajectory is planned using the most simplified dynamics model, and brings the simplified model close to the goal. When it is executed on the original model, the left half of the t-shirt does not fully cover the right half, which demonstrates the discrepancy between the simplified and original models. After one more iteration, a new simplified model is created with a small adjustment on the left side. A new trajectory is then planned, successfully manipulating the original model to the desired goal, which is displayed in Fig. 11(b).

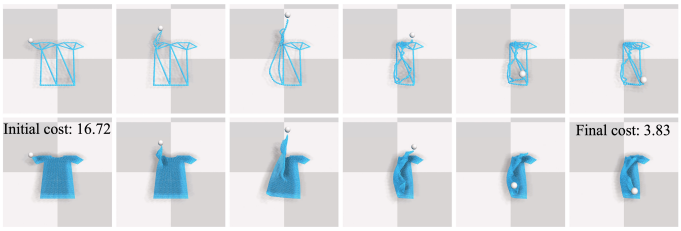
We also conduct experiments using RRT as the underlying planning algorithm with different dynamics models. A major difference between RRT and CEM implementations is that OMPL does not support a warm start. Therefore, we use the models in a manner similar to model predictive control: the state achieved by the previous plan serves as the initial state for the next planning loop, continuing until convergence.

From the results, the *Simplified Model* drops fastest for all tasks as well, and converges to better cost than *Original Model* and *Grid Model*, demonstrating the effectiveness of the proposed model simplification scheme. However, *Simplified & Original Model* with RRT behaves worse than CEM, which converges to worse cost than *Simplified Model* on cloth underneath folding task while obtaining similar costs for the other tasks. This reflects the defects of the altered warm start scheme, of which the latter planning loop is highly dependent on the former trajectory. The *Grid Model* gets the worst cost among all models, further underscoring the drawbacks of the modified warm start approach and indicating a lack of consistency in the planned trajectories for grid models of varying sizes. *Original Model* achieves acceptable results only for the cloth diagonal folding task, and converges to worse costs for other tasks due to the fact that the original model is more expensive and the state space is much larger compared to the simplified models.

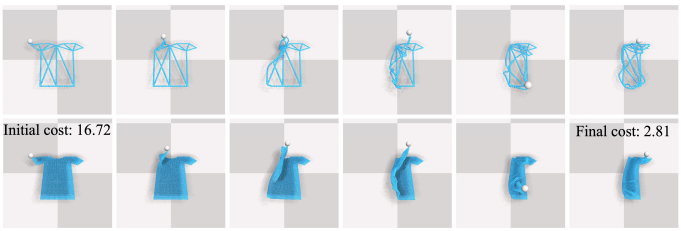
Regarding the iterative framework itself, we provide additional results on the average number of iterations required to achieve the best result. We also compare the trajectory cost obtained using the first single simplified model against the one found through the iterative framework, as shown in Tab. I. It is observed that while the first simplified model already yields near-satisfactory results, the iterative framework is still able to further optimize the trajectory, leading to improved performance.

E. Comparison with a Learned Model

To further verify the effectiveness of our proposed method, we compare it with a learning-based dynamics model originally developed by Lin et al. [44], called Visible Connectivity Dynamics (VCD). VCD employs a graph neural network



(a) First simplified model



(b) Second simplified model

Fig. 11. T-shirt side folding with simplified models. In image (a), a trajectory is planned using the initial simplified model, depicted in the first row, and then rolled out on the original model, shown in the second row. However, this does not achieve the intended goal, necessitating a new round of model simplification and motion planning. As demonstrated in image (b), these subsequent efforts successfully bring the cloth to the desired goal.

(GNN) trained to infer both the connections and the dynamics of the visible part of the deformable object. For training the GNN, they collect 2000 trajectories, each containing a single pick-and-place action, but decompose each trajectory into 100 smaller steps. The picking point is biased toward the highest point, while the placing location is selected randomly, making it more suitable for flattening tasks. After training, a single-step Random Shooting planner is used to iteratively generate pick-and-place actions towards a goal.

Besides, to measure the performance, they also introduced a metric named Normalized Improvement (NI), which quantifies the increase in the covered area relative to the maximum possible improvement. It is defined as $NI = \frac{s-s_0}{s_{max}-s_0}$, where s_0, s, s_{max} represent the initial, achieved, and maximum possible covered areas of the cloth, respectively.

We used the official implementation of VCD[1]. The original implementation uses visible point cloud data, which we call VCD(PC) here. Additionally, to ensure a fair comparison, we implemented a different version of VCD, where we made two key modifications. First, instead of using point cloud data, we directly used the ground-truth positions of the particles from the original model of the object. Second, we provided the actual edges explicitly, eliminating the need for the trained neural network to infer them. These modifications were made both to ensure fairness in comparison and to facilitate the precise specification of goal states for subsequent folding tasks. We call this modified version that uses *ground truth* particles and edges, VCD(GT).

For comparing our method against VCD, we used the validation set from Lin et al., which includes sampled cloth pieces with different sizes and initial configurations.

First, we attempted to compare our method and VCD, by using the same CEM planner that we used over the learned dynamics model. However, the VCD performed very poorly with CEM, because CEM tries to generate multi-action plans, which required running the VCD predictions over the outputs of itself, which resulted in accumulation of drastic drifts in the predicted states, already showing the limits of learned dynamics models.

Next, we compared our method to VCD, using the single-step Random Shooting planner used in their original implementation. To do this, we had to modify our method such that, after extracting a simplified model for the overall goal of the task, we used Random Shooting to plan an action. After the next state was reached with the execution of this action, we used this new state to extract a new simplified model (similar to line 11 in Alg. 1) and we combined this with the first simplified model extracted from the goal (similar to line 12 in Alg. 1).

The results for the cloth flattening task are shown in Fig. 12(a) using the NI metric. VCD(PC) and VCD(GT) achieve similar results, which are also very similar to the results reported in Lin et al., confirming VCD(GT)'s effectiveness. Our goal-informed simplified model achieves slightly better performance, despite requiring no training.

For the remaining folding tasks, we extended the definition of Normalized Improvement (NI) by replacing the coverage metric with the reward function (negative of the cost) specific to each task. Each task was evaluated for up to 20 pick-and-place steps, as further improvements beyond this were minimal. The results in Fig. 12 show that for diagonal folding, VCD achieves around 50% improvement. As task complexity increases, its performance degrades slightly for side folding and drops significantly for reflective folding. For cloth-underneath folding and T-shirt folding, the VCD fails to generate any feasible plans.

We hypothesize that for cloth-underneath folding, the model fails because it has not encountered similar state transitions during training. For T-shirt folding, the problem dimensionality is significantly higher than that of the training dataset, the shape is different, and the folding task was not specifically trained in the learned model.

The average planning time for each pick-and-place action is reported in Table II. For cloth folding tasks, the average planning time using VCD is approximately 70 seconds, which is consistent with the findings of Huang et al. [45], who reported a planning time of 100 seconds. In contrast, with our simplified dynamics model, the planner requires only 27 seconds per action, demonstrating a significant speed-up. For the T-shirt folding task, the increased number of particles leads to a substantial increase in the inference time of the learned dynamics model. This is further exacerbated by the fact that we did not apply downsampling to the particles or edge connections, preserving the original topology. However, our simplified model remains computationally efficient, requiring only a fraction of the time needed by the learned model.

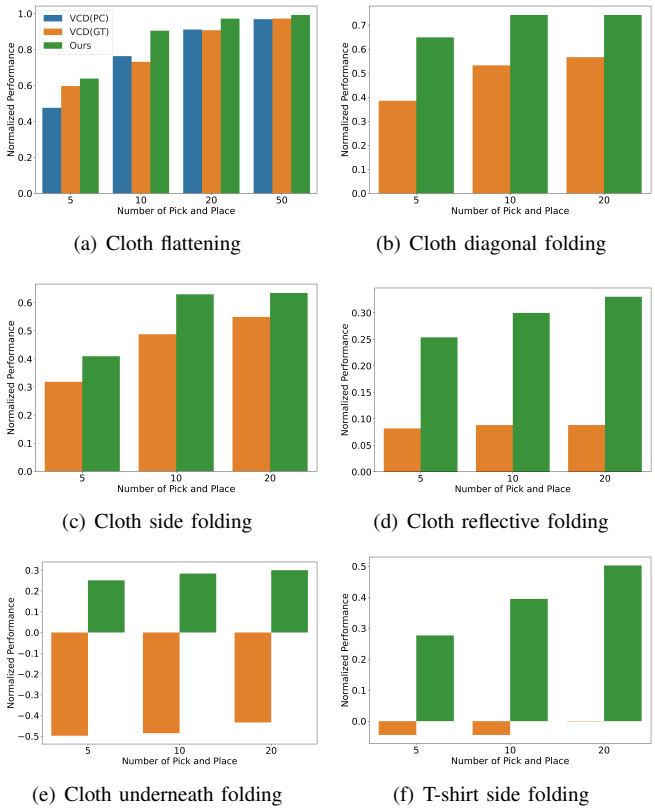
VI. Real-world Experiments

To verify the effectiveness of our proposed method in the real world and enhance the robustness of the execution, we build a closed-loop robotic experiment system composed of

[1]https://github.com/Xingyu-Lin/VCD

TABLE II
PLANNING TIME (S) FOR EACH PICK AND PLACE ACTION

	Flatten	Diagonal fold	Side fold	Reflective fold	Underneath fold	T-shirt fold
No. of particles	1720	1600	1849	1849	2025	3812
Time (VCD(GT))	70.6	70.2	70.8	70.4	71.0	608
Time (Ours)	27.4	27.3	27.4	27.3	27.5	28.2



(a) Cloth flattening (b) Cloth diagonal folding

(c) Cloth side folding (d) Cloth reflective folding

(e) Cloth underneath folding (f) T-shirt side folding

Fig. 12. Planner (Random Shooting) performance over number of actions (pick and place) executed for different tasks using VCD and our goal-simplified models.

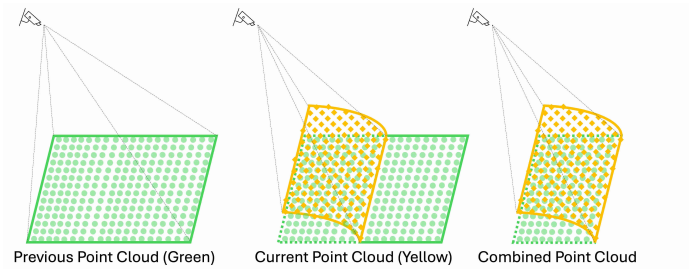


Fig. 13. The first image displays the initial point cloud in green. The second image shows the current point cloud in yellow, overlapping with the initial one, suggesting potential occlusion of some cloth points beneath the yellow surface. The third image presents the combined point cloud, which appears thicker due to this overlap. This augmentation of the point cloud aids the tracking algorithm in more accurately measuring the state of the cloth, especially given the penalization of movement in the underlying layers.

perception, planning, and control subsystems. The basic setup for the experiment includes a Franka Panda robot mounted on a stationary table, a flattened cloth (30 cm $\times$ 30 cm) randomly placed in front of it in various initial poses, a Realsense depth camera fixed in front of the robot. We also used an OptiTrack system to only calibrate the camera pose with respect to the robot, at the beginning of experiments.

For perception, we use a RealSense camera (extrinsically calibrated to the robot base) to track the shape of the cloth during manipulation. Specifically, we use CDCPD [60], [61] as the shape tracking method, but modify it in two ways to improve its performance for our tasks: 1) we explicitly reason about self-occlusions by comparing the distance between the current and previous point cloud, and combine them accordingly, the process of which is shown in Fig. 13; 2) instead of updating the positions of all tracking points at each time step, we retain the tracking point from previous time step if the point cloud around that area has not changed, while only optimizing the tracking points in areas where the point cloud has changed significantly.

For the dynamics model, two options are evaluated: one is to plan a coarse trajectory based on the simplified dynamics model and fine-tune it with the original dynamics model, which takes significant time but yields trajectories with lower cost; the second is to plan solely based on the most simplified dynamics model, which is much faster but gives slightly higher-cost trajectories.

For the control scheme, we first convert the motion plan to several pick-and-place motions according to the defined action space. When picking a particular point on the cloth, we use Moveit [62] to plan a valid trajectory to a certain height above the target using a position controller. After that, an impedance controller is switched on to gradually lower the gripper until it contacts the cloth. The use of impedance control is crucial, as it allows the gripper to apply a certain amount of pressure before grasping the cloth, preventing potential damage that could occur with a position-based controller.

Based on the established experiment system, we conduct experiments with cloth diagonal folding and side folding. We test both CEM and RRT based planners, and run each experiment with different dynamics models for 5 times, and calculate the success rate. The results are summarized in Table III, in which we achieve 100% success for the cloth diagonal folding task with various planner and model combinations. An experiment is considered successful if the robot completes all the actions in the planned trajectory and achieves a goal configuration of the cloth that meets user-defined criteria. For cloth side folding, which requires more alternations between different picking points, only CEM achieves 80% success with the *Simplified & Original Model*, and 60% success with the *Simplified Model*. RRT fails to accomplish the task, because the planned trajectories involve a much higher number of re-grasps. Among all the failure cases in real-world experiments, the main reasons include: 1) failure of the tracking system due

TABLE III
REAL-WORLD EXPERIMENT RESULTS

Task	Planner	Dynamics model	Planning time (min)	Success Rate	Average picks
Cloth Diagonal Folding	CEM	Simplified & Original	15.08	5/5	4.20
		Simplified	2.07	5/5	4.80
	RRT	Simplified & Original	10.00	5/5	7.00
		Simplified	1.00	5/5	8.00
Cloth Side Folding	CEM	Simplified & Original	21.26	4/5	5.60
		Simplified	3.01	3/5	6.40
	RRT	Simplified & Original	10.00	0/5	18.20
		Simplified	1.00	0/5	18.60

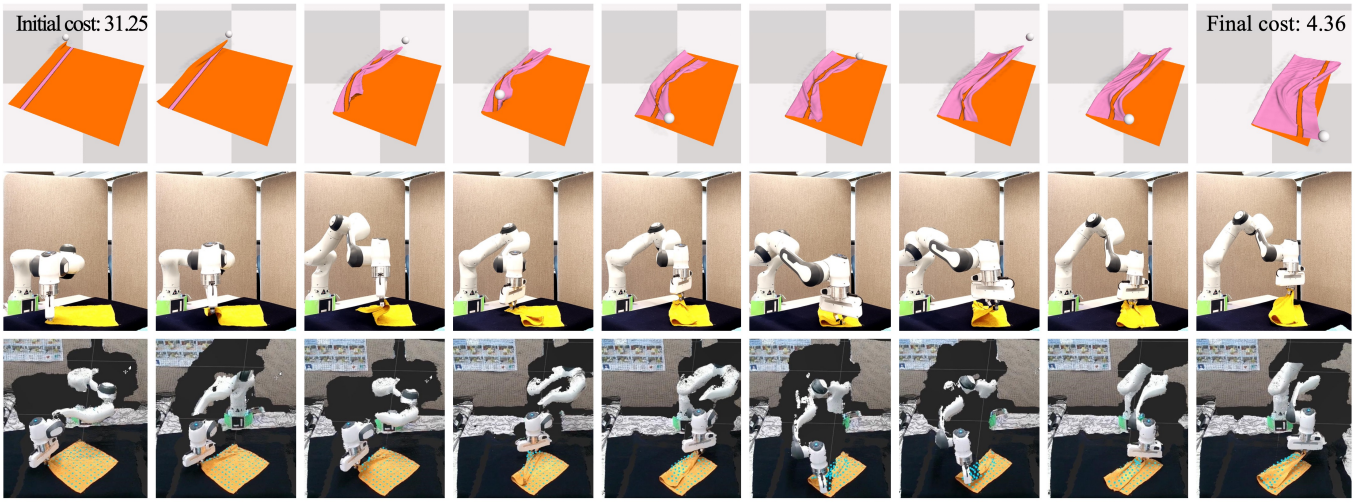


Fig. 14. Real-world experiment for cloth side folding with fine models. The first row displays the planned trajectory on the original model. The second row captures the robot executing this trajectory. The third row shows the tracking results.

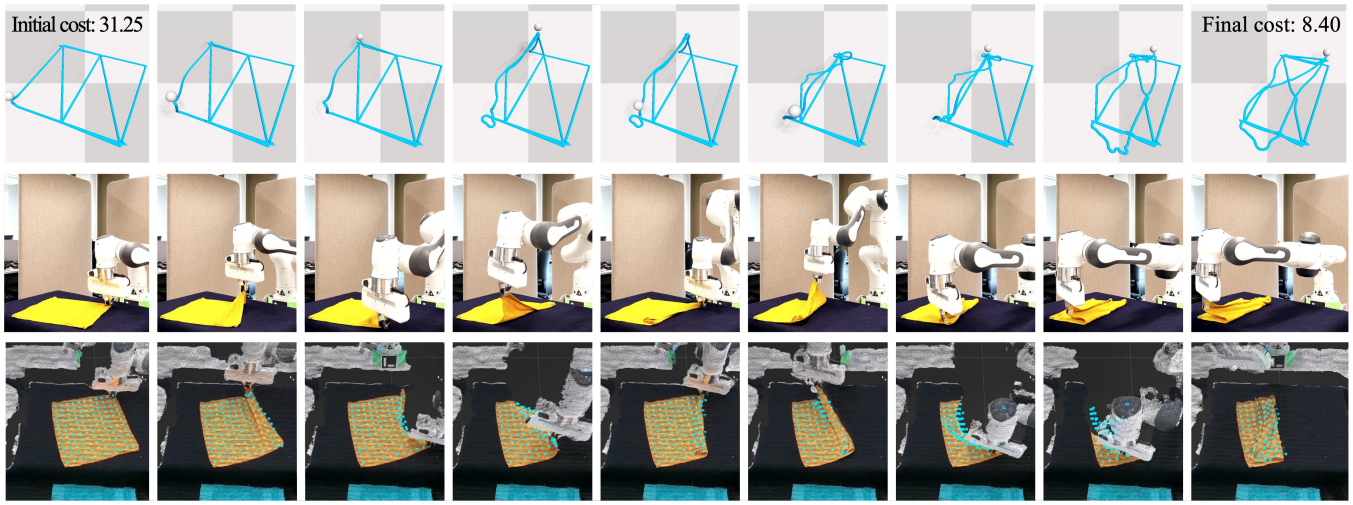


Fig. 15. Real-world experiment for cloth side folding with simplified models. The first row illustrates the planned trajectory using the simplified model. The second row demonstrates the trajectory rollout on the original model. The third row captures the robot executing this trajectory, and the fourth row displays the tracking results.

to severe occlusion by the robot arm; 2) the gripper picking an incorrect position due to tracking errors; 3) the gripper picking multiple layers of the cloth; and 4) the picking position being out of the robot's workspace, causing joint limit violations.

We also evaluate the performance of the real robot experiments by reconstructing the cloth's final state (as reported by our CDCPD-based shape tracking method) in simulation and assessing its similarity to the goal state using two metrics: NI and Intersection over Union (IoU). We considered an experiment as successful if either NI or IoU exceeds 70. For the diagonal folding tasks, the average successful NI is 0.85 with standard deviation 0.07, and the IoU is 0.92 with standard deviation 0.04. For the side folding tasks, the average successful NI is 0.69 with standard deviation 0.12, and the IoU is 0.87 with standard deviation 0.06.

Images of the real robot folding the cloth sideways with CEM as the underlying motion planner are shown in Fig. 14 and Fig. 15. Please refer to the attached video to view the entire manipulation process for both diagonal folding and side folding tasks. The Panda robot successfully achieves both tasks. Within each task, the gripper can re-grasp the cloth showing that our proposed controlling scheme for executing the plan is effective and the planned trajectory in simulation can be executed successfully to achieve the desired cloth folding. During execution, if the behavior of the cloth diverges from the predictions made during simulation, the robot can adaptively modify its actions based on the real-time perceived state of the cloth. This adaptability demonstrates significantly improved robustness compared to open-loop execution, where adjustments to unforeseen changes are not possible.

VII. Conclusion

We have proposed a framework of reducing the action space and iteratively simplifying the dynamics model given a desired goal state for deformable object manipulation. Two workflows of geometric model simplification are developed for 1-D linear and 2-D surface objects respectively. We further reduce the action space and simplify the dynamics model to accelerate motion planning for deformable object manipulation. Extensive simulation and real robot experiments are conducted, which demonstrate that our proposed method can improve the efficiency and performance of a motion planner. Besides, our approach allows for faster trajectory planning for various deformable manipulation tasks and can adaptively refine the model to achieve better results. However, our method has certain limitations. One notable limitation is the assumption that the initial state of the deformable object is flattened, which may not always hold in practical scenarios. Additionally, we recognize the importance of tracking systems for precise manipulation, state estimation, and bridging the gap between simulation and reality. Another challenge is that not all goal states can be explicitly specified—for instance, vision-based or language-based goal specifications may be difficult to interpret using our approach. While our framework significantly boosts efficiency, the computational cost of motion planning remains relatively high, limiting its applicability in time-critical or highly dynamic environments. An exciting avenue for future research involves incorporating the efficient simplified dynamics model for deformable object shape tracking. Moreover, we can also generate a batch of simplified models simultaneously, whether goal-conditioned or non-goal-conditioned, to predict which model offers the most improvement for a specific task, and intelligently utilize these models until a satisfactory trajectory is found or the process times out.

References

[1] Z. Huang, X. Lin, and D. Held, "Self-supervised cloth reconstruction via action-conditioned cloth tracking," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7111–7118.

[2] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robot Learning*. PMLR, 2022, pp. 24–33.

[3] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, "Speedfolding: Learning efficient bimanual folding of garments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1–8.

[4] L. Yang, L. Yang, H. Sun, Z. Zhang, H. He, F. Wan, C. Song, and J. Pan, "One fling to goal: Environment-aware dynamics for goal-conditioned fabric flinging," *arXiv preprint arXiv:2406.14136*, 2024.

[5] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep transfer learning of pick points on fabric for robot bed-making," in *The International Symposium of Robotics Research*. Springer, 2019, pp. 275–290.

[6] X. Li, X. Su, and Y.-H. Liu, "Vision-based robotic manipulation of flexible pcbs," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2739–2749, 2018.

[7] E. Torgerson and F. W. Paul, "Vision-guided robotic fabric manipulation for apparel manufacturing," *IEEE Control Systems Magazine*, vol. 8, no. 1, pp. 14–20, 1988.

[8] A. Longhini, Y. Wang, I. Garcia-Camacho, D. Blanco-Mulero, M. Moletta, M. Welle, G. Alenyà, H. Yin, Z. Erickson, D. Held *et al.*, "Unfolding the literature: A review of robotic cloth manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, 2024.

[9] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.

[10] Y. Deng, C. Xia, X. Wang, and L. Chen, "Deep reinforcement learning based on local gnn for goal-conditioned deformable object rearranging," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1131–1138.

[11] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme, "Learning deformable object manipulation from expert demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8775–8782, 2022.

[12] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[13] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, "Modeling of deformable objects for robotic manipulation: A tutorial and review," *Frontiers in Robotics and AI*, vol. 7, p. 82, 2020.

[14] Y. Bai, W. Yu, and C. K. Liu, "Dexterous manipulation of cloth," in *Computer Graphics Forum*. Wiley Online Library, 2016, pp. 523–532.

[15] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 67–77, 2022.

[16] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip, "Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3964–3971, 2023.

[17] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[18] J. Borras, G. Alenya, and C. Torras, "A grasping-centered analysis for cloth manipulation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 924–936, 2020.

[19] 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*. IEEE, 2010, pp. 2308–2315.

[20] Y. Qiu, J. Zhu, C. Della Santina, M. Gienger, and J. Kober, "Robotic fabric flattening with wrinkle direction detection," in *International Symposium on Experimental Robotics*. Springer, 2023, pp. 339–350.

[21] M. Ruan, D. McConachie, and D. Berenson, "Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 512–519.

[22] D. McConachie, A. Dobson, M. Ruan, and D. Berenson, "Manipulating deformable objects by interleaving prediction, planning, and control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 957–982, 2020.

[23] S. Wang, R. Papallas, M. Leonetti, and M. Dogar, "Goal-conditioned action space reduction for deformable object manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3623–3630.

[24] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robotics and computer-integrated manufacturing*, vol. 28, no. 2, pp. 154–163, 2012.

[25] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 987–993.

[26] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6000–6006.

[27] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8803, 2021.

[28] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, p. 4, 2019.

[29] L. Sun, G. Aragon-Camarasa, P. Cockshott, S. Rogers, and J. P. Siebert, "A heuristic-based approach for flattening wrinkled clothes," in *Towards Autonomous Robotic Systems: 14th Annual Conference, TAROS 2013, Oxford, UK, August 28–30, 2013, Revised Selected Papers 14*. Springer Berlin Heidelberg, 2014, pp. 148–160.

[30] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.

[31] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *Conference on Robot Learning*. PMLR, 2021, pp. 564–574.

[32] T. Lips, V.-L. De Gusseme *et al.*, "Learning keypoints for robotic cloth manipulation using synthetic data," *IEEE Robotics and Automation Letters*, 2024.

[33] P. Zhou, J. Zhu, S. Huo, and D. Navarro-Alarcon, "Lasesom: A latent and semantic representation framework for soft object manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5381–5388, 2021.

[34] S. Arnold, D. Tanaka, and K. Yamazaki, "Cloth manipulation planning on basis of mesh representations with incomplete domain knowledge and voxel-to-mesh estimation," *Frontiers in Neurorobotics*, vol. 16, p. 1045747, 2023.

[35] X. Ma, D. Hsu, and W. S. Lee, "Learning latent graph dynamics for deformable object manipulation," *arXiv preprint arXiv:2104.12149*, vol. 2, 2021.

[36] C. Li, Z. Ai, T. Wu, X. Li, W. Ding, and H. Xu, "Deformnet: Latent space modeling and dynamics prediction for deformable object manipulation," *arXiv preprint arXiv:2402.07648*, 2024.

[37] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2341–2353, 2022.

[38] D. McConachie, T. Power, P. Mitrano, and D. Berenson, "Learning when to trust a dynamics model for planning in reduced state spaces," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3540–3547, 2020.

[39] D. McConachie and D. Berenson, "Bandit-based model selection for deformable object manipulation," in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer International Publishing, 2020, pp. 704–719.

[40] T. Power and D. Berenson, "Keep it simple: Data-efficient learning for controlling complex systems with simple models," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1184–1191, 2021.

[41] P. Zhou, P. Zheng, J. Qi, C. Li, C. Yang, D. Navarro-Alarcon, and J. Pan, "Bimanual deformable bag manipulation using a structure-of-interest based latent dynamics model," *arXiv preprint arXiv:2401.11432*, 2024.

[42] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for multi-step, multi-task fabric manipulation," *arXiv preprint arXiv:2003.09044*, 2020.

[43] Z. Weng, F. Paus, A. Varava, H. Yin, T. Asfour, and D. Kragic, "Graph-based task-specific prediction models for interactions between deformable and rigid objects," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5741–5748.

[44] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning visible connectivity dynamics for cloth smoothing," in *Conference on Robot Learning*. PMLR, 2022, pp. 256–266.

[45] Z. Huang, X. Lin, and D. Held, "Mesh-based dynamics with occlusion reasoning for cloth manipulation," *arXiv preprint arXiv:2206.02881*, 2022.

[46] P. Mitrano, A. LaGrassa, O. Kroemer, and D. Berenson, "Focused adaptation of dynamics models for deformable object manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5931–5937.

[47] T. M. Lee, Y. J. Oh, and I.-K. Lee, "Efficient cloth simulation using miniature cloth and upscaling deep neural networks," *arXiv preprint arXiv:1907.03953*, 2019.

[48] Y.-M. Chen and M. Posa, "Optimal reduced-order modeling of bipedal locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8753–8760.

[49] S. Tonkens, J. Lorenzetti, and M. Pavone, "Soft robot optimal control via reduced order finite element models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 010–12 016.

[50] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-anything: Automating real-world robots using vision-language models," *arXiv preprint arXiv:2406.18915*, 2024.

[51] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen *et al.*, "Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation," *IEEE Robotics and Automation Letters*, 2025.

[52] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 209–216.

[53] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia *et al.*, "Meshlab: an open-source mesh processing tool." in *Eurographics Italian chapter conference*, vol. 2008. Salerno, Italy, 2008, pp. 129–136.

[54] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," in *Proceedings. IEEE international conference on multimedia and expo*, vol. 1. IEEE, 2002, pp. 705–708.

[55] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 432–448.

[56] D. J. Balkcom and M. T. Mason, "Robotic origami folding," *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 613–627, 2008.

[57] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, pp. 19–67, 2005.

[58] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: a survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[59] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[60] Y. Wang, D. McConachie, and D. Berenson, "Tracking partially-occluded deformable objects while enforcing geometric constraints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 199–14 205.

[61] C. Chi and D. Berenson, "Occlusion-robust deformable object tracking without physics simulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6443–6450.

[62] S. Chitta, "Moveit!: an introduction," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 3–27, 2016.

Shengyin Wang received his Ph.D. in Robotics from the University of Leeds in 2025. He obtained his B.Eng. and M.Eng. degrees in Aeronautics and Astronautics from the Beijing Institute of Technology in 2017 and 2020, respectively. His research interests lie at the intersection of robotic manipulation, motion planning, and deformable object handling. His work focuses on developing efficient algorithms for complex robotic tasks, particularly model-based planning, dynamics learning, and manipulation of non-rigid materials.

Matteo Leonetti received his PhD from Sapienza University of Rome in 2011. He is currently a Senior Lecturer in Computer Science at King's College London. He was a Lecturer at the University of Leeds between 2016-2021, a post-doctoral researcher at the University of Texas at Austin between 2013-2015, and a post-doctoral researcher at the Italian Institute of Technology between 2012-2013. He is a member of the editorial board of the Artificial Intelligence Journal. His research focuses on reinforcement learning and autonomous service robots.

Mehmet Dogar received his PhD from the Robotics Institute at the Carnegie Mellon University in 2013. He is currently a Professor in Robotics and AI at the School of Computer Science, University of Leeds, UK. He was a Visiting Professor at ETH Zurich in 2023 and a post-doctoral researcher at CSAIL, MIT between 2013-2015. Prof. Dogar is an Associate Editor for IEEE Transactions on Robotics and also for the International Journal of Robotics Research. His research focuses on robotic object manipulation.