# Interleaving Planning and Control for Deformable Object Manipulation

Dale M$^{c}$Conachie, Mengyao Ruan, and Dmitry Berenson

**Key words:** motion planning, deformable object manipulation

**Abstract** We present a framework for deformable object manipulation that interleaves planning and control, enabling complex manipulation tasks without relying on high-fidelity modeling or simulation. The key question we address is when should we use planning and when should we use control to achieve the task? Planners are designed to find paths through complex configuration spaces, but for highly underactuated systems such as deformable objects achieving a specific configuration is very difficult even with high-fidelity models. Conversely, controllers can be designed to achieve specific configurations, but they can be trapped in undesirable local minima due to obstacles. Our approach consists of three components: (1) A global motion planner to generate gross motion of the deformable object; (2) A local controller for refinement of the configuration of the deformable object; and (3) A novel deadlock prediction algorithm to determine when to use planning versus control. By separating planning from control we are able to use different representations of the deformable object, reducing overall complexity and enabling efficient computation of motion. We demonstrate that our framework is able to successfully perform several manipulation tasks in simulation which cannot be performed using either our controller or planner alone.

## 1 Introduction

Examples of deformable object manipulation range from domestic tasks like folding clothes to time and safety critical tasks such as robotic surgery. One of the challenges in planning for deformable object manipulation is the high number of degrees of freedom involved; even approximating the configuration of a piece of cloth with a $4 \times 4$ grid results in a 48 degree of freedom configuration space. In addition, the dynamics of the deformable object are difficult to model [8]; even with high-fidelity modeling and simulation, planning for an individual task can take hours [2]. Local controllers on the other hand are able to very efficiently generate motion, however, they are only able to successfully complete a task when the initial configuration is favorable [3, 19].

The central question we address in this work is how can we combine the strengths of global planning with the strengths of local control while mitigating the weak-

University of Michigan, Ann Arbor, MI, USA, 48109. e-mail: {dmcconac,ruanmiao}@umich.edu, berenson@eecs.umich.edu. This work was supported in part by NSF grant IIS-1656101 and ONR grant N000141712050.

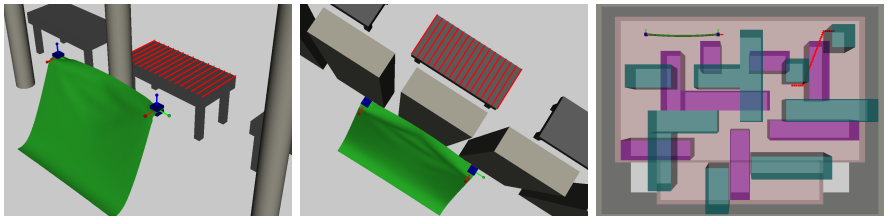

**Fig. 1** Three example manipulation tasks for our framework. In the first two tasks, the objective is to cover the surface of the table (indicated by the red lines) with the cloth (shown in green). In the first task, the grippers (shown in blue) can freely move however the cloth is obstructed by a pillar. In the second task, the grippers must pass through a narrow passage before the table can be covered. In the third task, the robot must navigate a rope (shown in green in the top left corner) through a three-dimensional maze before covering the red points in the top right corner. The maze consists of top and bottom layers (purple and green, respectively). The rope starts in the bottom layer and must move to the target on the top layer through an opening (bottom left or bottom right).

ness of each? We propose a framework for interleaving planning and control which uses global planning to generate gross motion of the deformable object, and a local controller to refine the configuration of the deformable object within the local neighborhood. By separating planning from control we are able to use different representations of the deformable object, each suited to efficient computation for their respective roles. In order to determine when to use each component, we introduce a novel deadlock prediction algorithm that is inspired by topologically based motion planning methods [4, 13]. By answering the question "Will the local controller get stuck?" we can predict if the local controller will be unable to achieve the task from the current configuration. If we predict that the controller will get stuck we can then invoke the global planner, moving the deformable object into a new neighbourhood from which the local controller may be able to succeed. The key to our efficient prediction is forward-propagating only the stretching constraint, assuming the object will comply to contact.

We seek to solve problems where we need to arrange the object in a particular way (e.g. covering a table with a tablecloth) but where there is also complex environment geometry preventing us from directly completing the task. While we can't claim to solve all problems in this class (in particular in environments where the deformable object can get snagged), we can solve an important sub-class; in this work we restrict our focus to controllers of the form described in Sec. 4.1, and tasks suited to these controllers. Examples of these types of tasks are shown in Fig. 1. In our experiments we show that this iterative method of interleaving planning and control is able to successfully perform several interesting tasks for one-dimensional and two-dimensional deformable objects (i.e. rope and cloth) where our planner or controller alone are unable to succeed.

## 2 Related Work

Robotic manipulation of deformable objects has been studied in many contexts ranging from surgery to industrial manipulation (see [16] for an extensive survey).

Motion planning for manipulation of deformable objects is an active area of research [14]. Saha *et al.* [26] present a Probabilistic Roadmap (PRM [15]) that plans for knot-tying tasks with rope. Rodriguez *et al.* [24] study motion planning in fully deformable simulation environments. Their method, based on Rapidly-exploring Random Trees (RRTs [18]), applies forces directly to an object to move it through narrow spaces while using the simulator to compute the resulting deformations. Recently Frank *et al.* [9] presented a method that pre-computes deformation simulations in a given environment to enable fast multi-query planning. Other sampling-based approaches [1, 6, 10, 17, 20, 25] rely on being able to explicitly define the goal region for the deformable object, whereas our approach uses an error-function-based task definition. Also, the above methods rely on potentially time-consuming physical simulation of the deformable object, while our method does not.

Model-based visual servoing approaches [11, 28, 30] bypass planning entirely, and instead use a local controller to determine how to move the robot end-effector for a given task. Our recent work [3, 19] as well as [22] bypass the need for an explicit deformable object model, instead using Jacobian approximation controllers to drive the deformable object to the attractor of the starting state. Rather than using just a planner or just a controller, our framework uses both components, each when most useful.

Approaches based on learning from demonstration [12, 27] avoid planning and deformable object modelling challenges entirely by using offline demonstrations to teach the robot specific manipulation tasks; however, when a new task is attempted a new training set needs to be generated. In our application we are interested in a way to manipulate a deformable object without a high-fidelity model or training set available *a priori*. For instance, imagine a robot encountering a new piece of clothing for a new task. While it may have models for previously-seen clothes or training sets for previous tasks, there is no guarantee that those models or training sets are appropriate for the new task.

Our planning method has some similarity to topological [4, 13] and tethered robot [5, 29] planning techniques; these methods use the topological structure of the space to define homotopy classes, either as a direct planning goal, or as a way to help inform planning in the case of tethered robots. Planning for some deformable objects, in particular rope or string, can be viewed as an extension of the tethered robot case where the base of the tether can move. This extension, however, requires a very different approach to homotopy than is traditionally used, particularly when working in three-dimensional space instead of a planar environment. In our work we use *visiblity deformations* from [13] as a way to encode homotopy-like classes of configurations.

## 3 Problem Statement

Let the robot be represented by a set of $G$ grippers with configuration $q \in SE(3)^G$. We assume that the robot configuration can be measured exactly. In this work we assume the robot to be a set of free floating grippers; in practice we can track the motion of these with inverse kinematics. We use the Lie algebra [21] of $SE(3)$ to

represent robot gripper velocities. This is the tangent space of $SE(3)$, denoted as $\mathfrak{se}(3)$. The velocity of a single gripper $g$ is then $\dot{q}_g = \begin{bmatrix} v_g^T & \omega_g^T \end{bmatrix}^T \in \mathfrak{se}(3)$ where $v_g$ and $\omega_g$ are the translational and rotational components of the gripper velocity. We define the velocity of the entire robot to be $\dot{q} = \begin{bmatrix} \dot{q}_1^T & \dots & \dot{q}_G^T \end{bmatrix}^T \in \mathfrak{se}(3)^G$.

The configuration of a deformable object is a set $\mathcal{P} \subset \mathbb{R}^3$ of $P = |\mathcal{P}|$ points. We assume that we have a method of sensing $\mathcal{P}$. We define a task based on a set of $T$ target points $\mathcal{T} \subset \mathbb{R}^3$, a function $\rho$ which measures the alignment error between $\mathcal{P}$ and $\mathcal{T}$, and a termination condition $\Omega$. Let a robot controller be a function $C(q, \mathcal{P}, \mathcal{T}, \rho, \Omega)$ which maps the system state $(q, \mathcal{P})$ and task $(\mathcal{T}, \rho, \Omega)$ to a desired robot motion $\dot{q}_{cmd}$. In this work we restrict our discussion to tasks and controllers of the form introduced in our previous work [3, 19]; these controllers are local, i.e. at each time $t$ they choose an incremental movement $\dot{q}_{cmd}$ which reduces the alignment error as much as possible at time $t+1$.

Let $E(\dot{q}_{cmd}, q, \mathcal{P}) = \dot{q}_{act}$ be the true motion of the robot when $\dot{q}_{cmd}$ is executed for unit time; in this work we will be predicting the future state of the system, thus it is not sufficient to consider only $\dot{q}_{cmd}$, we must also consider $\dot{q}_{act}$. Modelling inaccuracies as well as manipulation in contact can lead to meaningful differences between $\dot{q}_{cmd}$ and $\dot{q}_{act}$.

The problem we address in this work is how to find a sequence of $N_e$ robot commands $\{\dot{q}_{cmd,1}, \dots, \dot{q}_{cmd,N_e}\} = \dot{Q}_{cmd}$ such that each motion is feasible, i.e. it should not bring the grippers into collision with obstacles, should not cause the object to stretch excessively, and should not exceed the robot's maximum velocity $\dot{q}_{\max}$. Let these feasibility constraints be represented by $A(\dot{q}) = 0$. Then the problem we seek to solve is:

$$\begin{aligned} \text{find} \quad & \dot{Q}_{cmd} \\ \text{s.t.} \quad & \Omega(\mathcal{P}_{N_e}) = \texttt{true} \\ & A(\dot{q}_{act,t}) = 0,\ t = 1, \dots, N_e \end{aligned} \tag{1}$$

where $\mathcal{P}_{N_e}$ is the configuration of the deformable object after executing $\dot{Q}_{cmd}$.

Solving this problem directly is impractical in the general case for two major reasons. First, modeling a deformable object accurately is very difficult in the general case, especially if it contacts other objects or self-collides. Second, even given a perfect model, computing precise motion of the deformable object requires FEM simulation and is very time consuming. We seek a method which does not rely on high-fidelity modelling and simulation; instead we present an incremental solution which leverages the strengths of both global planning and local control in order to efficiently perform the task. This paper focuses on the case of using two grippers, but our method could be extended to the multi-gripper case.

## 4 Interleaving Planning and Control

Global planners are effective at finding paths through complex configuration spaces, but for highly underactuated systems such as deformable objects achieving a specific configuration is very difficult even with high-fidelity models; this means that we cannot rely on them to complete a task independent of a local controller. In or-

der for the local controller to complete the task, the system must be in the correct basin of attraction. From this point of view it is not the planner's responsibility to complete a task but rather to move the system into the right basin for the local controller to finish the task. By explicitly separating planning from control we can use different representations of the deformable object for each component; this allows us to use a highly-simplified model of the deformable object for global planning to generate gross motion of the deformable object, while using an independent local approximation for the controller. The key question then is when should we use global planning versus local control? We answer this question by considering the limitations of a given controller.

### *4.1 Local Control*

For our local controller we use a controller of the form introduced in [3, 19]. These controllers locally minimize error $\rho$ while obeying the constraints of the system. For every target point $\mathcal{T}_i \in \mathcal{T}$ we define a potential field pointing towards $\mathcal{T}_i$ using Dijkstra's algorithm. This gives us the shortest collision-free path between any point in the workspace and the target point, as well as the distance travelled along that path. These potential fields are used to define the best direction to manipulate the deformable object in order to locally reduce error as much as possible at each timestep. This error reduction term is then combined with stretching avoidance and gripper collision avoidance methods to generate robot motion commands while obeying velocity constraints. Full details can be found in [19].

One limitation of this approach is that the individual potential fields are defined and applied independently of each other; this means that the potential fields that are combined to define the direction to move the deformable object can cause the controller to move the grippers on opposite sides of an obstacle, leading to poor local minima. Examples of this scenario are shown in Figures 2, 4, and 5. In addition, while this local controller can avoid obstacles, it does not explicitly have any ability to *go around* obstacles as shown in Figure 3. In order to address these limitations we introduce a novel deadlock prediction algorithm to detect when the system $(q_t, \mathcal{P}_t)$ is in a state that will lead to deadlock if we continue to use the local controller.

### *4.2 Predicting Deadlock*

Predicting deadlock is important for two reasons; first we don't want to waste time executing motions that will not achieve the task. Second, we want to avoid the computational expense of planning our way out of a cul-de-sac. By predicting deadlock before it happens we address both of these concerns.

We consider a controller to be deadlocked if the commanded motion produces (nearly) no actual motion, and the task termination condition is not met:

$$\begin{aligned} \dot{q}_{act,t} &\approx 0 \\ \Omega(\mathcal{P}_t) &= \texttt{false}. \end{aligned} \tag{2}$$

**Algorithm 1** PredictDeadlock$(q_t, \mathcal{P}_t, B_t, \text{Path}, \mathcal{T}, L_{max}, N_p)$

ConfigurationHistory $\leftarrow$ [ConfigurationHistory, $q_t$]
ErrorHistory $\leftarrow$ [ErrorHistory, $\rho(\mathcal{P}_t)$]
**for** $n = 0, \ldots, N_p - 1$ **do**
  **if** $Path \neq \emptyset$ **then**
    $\mathcal{P}_{t+n+1} \leftarrow$ FollowVectorField$(\mathcal{P}_{t+n}, \mathcal{T})$
    $\dot{q}_{cmd,t+n+1} \leftarrow C_{simple}(q_{t+n}, \mathcal{P}_{t+n}, \mathcal{T}, \rho, \Omega)$
    $q_{t+n+1} \leftarrow q_{t+n} + \dot{q}_{cmd,t+n}$
  **else**
    $q_{t+n+1} \leftarrow q_{t+n}+$ FollowPath(Path)
  **end if**
  $B_{t+n+1} \leftarrow$ ForwardPropagateBand$(B_{t+n}, q_{t+n+1})$
**end for**
**if** PredictOverstretch$(L_{max})$ **or** NoProgress() **then**
  **return** `true`
**else**
  **return** `false`
**end if**

In general we cannot predict if the system will get stuck in the limit; our controllers and the system itself are complex, and as such we do not have a good way to evaluate the attractor of the current state. Instead we predict if the system will get stuck within a prediction horizon $N_p$ timesteps. We divide our deadlock prediction algorithm into three parts and discuss each in turn: 1) estimating gross motion; 2) predicting overstretch; and 3) progress detection.

#### 4.2.1 Estimating Gross Motion

The idea central to our prediction (Alg. 1) is that while we may not be able to determine precisely how a given controller will steer the system, we can capture the gross motion of the system and estimate if the controller will be deadlocked. We split the prediction into two parts; first we assume that controller $C$ is able to manipulate the deformable object with a reasonable degree of accuracy within a local neighbourhood of the current state. This allows us to approximate the motion of the deformable object by following the task defined potential fields for each $\mathcal{P}_i \in \mathcal{P}$. Examples of this approximation are shown in Figures 2-5. Next we use a simplified version of $C$ which omits the stretching avoidance term to predict the commands sent to the robot. This term is omitted as it can be sensitive to the exact configuration of the deformable object, which is not considered in this approximation. If we are executing a path then we can use the planned path directly to predict overstretch.

#### 4.2.2 Predicting Overstretch

Next we introduce the notion of a *virtual elastic band* between the grippers. This virtual elastic band is based on Quinlan's path deformation algorithm [23] and is used both in deadlock prediction as well as global planning (Sec. 4.3) to determine if a given gripper trajectory will stretch the deformable object beyond a task specified maximum stretching factor $\lambda$. This is designed to detect when the deformable object can get caught on an obstacle, preventing forward motion of the grippers. At each

timestep the elastic band is initialized with the shortest path between the grippers through the deformable object, and then "pulled" tight using the internal contraction force described in [23] and a hard constraint for collision avoidance. The endpoints of the band track the predicted translation of the grippers. This band represents the constraint that must be satisfied for the object not to tear. By considering only this constraint on the object in prediction, we are implicitly relying on the object to comply to the contacts as it is moved by the grippers. We discuss the limitations of this assumption in the discussion (Sec. 6).

Let $L_{t+n}$ be the length of the path defined by the virtual elastic band $B_{t+n}$ at timestep $n$ in the future, and $L_{max}$ be the longest allowable band length. To use this length sequence to predict if the controller will overstretch the deformable object, we perform three filtering steps: an annealing low-pass filter, a filter to eliminate cases where the band is in freespace, and the detector itself which predicts overstretch. We use a low-pass annealing filter with annealing constant $\alpha \in [0,1)$ to mitigate the effect of numerical and approximation errors which could otherwise lead to unnecessary planning:

$$\begin{aligned} \tilde{L}_{t+1} &= L_{t+1} \\ \tilde{L}_{t+n} &= \alpha \tilde{L}_{t+n-1} + (1-\alpha) L_{t+n} \quad , n = 2, \ldots, N_p \quad . \end{aligned} \tag{3}$$

Second, we discard from consideration any bands which are not in contact with an obstacle; we can eliminate these cases because our local controller includes an overstretch avoidance term which will prevent this case in general. Last we compare the filtered length of any remaining band predictions to $L_{max}$; if after filtering, there is a estimated band length $\tilde{L}$ that is larger than $L_{max}$ then we predict that the local controller will be stuck. An example of this type of detection is shown in Figure 2, where the local controller will wrap the cloth around the pole, eventually becoming deadlocked in the process.

#### 4.2.3 Progress Detection

Last, we track the progress of the grippers and task error to estimate if the controller $C$ is making progress towards the task goal. This is designed to detect cases when the grippers themselves are trapped against an obstacle. Naively we could look for instances when $\dot{q}_{act} = 0$ however due to sensor noise, actuation error, and using discrete math in a computer, we need to use some threshold instead. At the same time we want to avoid false positives, where the grippers are moving slowly however task error is decreasing. To address these concerns we introduce three parameters: history window $N_h$, error improvement threshold $\beta_e$, and configuration distance threshold $\beta_m$. If over the last $N_h$ timesteps, the improvement in error is less than $\beta_e$, and the grippers have moved less than $\beta_m$, then we predict that the controller will not be able to reach the goal from the current state and trigger global planning.

### *4.3 Global Planning*

In order to enable efficient planning, we need to approximate the configuration of the deformable object in a way that captures the gross motion of the deformable ob-

ject without being prohibitively expensive to use. We use the same approach from Sec. 4.2.2, but the interpretation in this use is slightly different; the virtual elastic band is a proxy for the leading edge of the deformable object. In this way we can plan to move the deformable object to a different part of the workspace without needing to simulate the entire deformable object, instead the deformable object conforms to the environment naturally.

We also do not consider the orientation of the grippers while planning, instead treating the grippers as spheres. The resulting space that we are planning in is $\mathbb{R}^{3G} \times V$, where $V$ represents the space of virtual elastic band configurations. For our global planner we use a conventional RRT-Connect approach as the central planning mechanism (Alg. 2). We cannot explicitly sample $V$ as the state of the band depends on the path the grippers take, hence we sample only the position of the grippers, forward-propagating the virtual elastic band from the nearest neighbour towards the sampled gripper positions until the grippers collide with an obstacle, or the length of the virtual elastic band $L$ exceeds $L_{max}$.

In order to make progress towards achieving the task, we want to set the goal for the RRT to be a configuration that we have not explored with the local controller. We do so in two parts; we find the set of all target points $\mathcal{T}_U$ which are contributing to task error, split these points into two clusters, and use the cluster centers to define the goal position of the grippers, $q_{xyz,goal}$. Second, we set the goal configuration of the virtual elastic band to be any configuration that is not similar to a *blacklist* of virtual rubber bands. This blacklist is the set of all band configurations from which we predicted that the local controller would be deadlocked in the future (Sec. 4.2).

To define similarity we use Jaillet and Siméon's *visibility deformation* [13] definition to compare two virtual elastic bands. Intuitively two virtual elastic bands are similar if you can sweep a straight line connecting the two bands from the start points to the end points of the two bands without intersecting an obstacle. Unlike the original use, we do not constrain the start and end points of each path to match, but the algorithm is identical. We use this as a heuristic to find states that are dissimilar from states where we have already predicted that the local controller would be deadlocked. Let $\text{VisCheck}(B, \text{Blacklist}) \rightarrow \{0,1\}$ denote this visibility deformation check, returning 1 if $B$ is similar to a band in the blacklist and 0 otherwise. Let $R_{goal} = \{(q_{xyz,goal}, B_{goal}) \mid \text{VisCheck}(B_{goal}, \text{Blacklist}) = 0\}$ be the goal region defined by the gripper target above.

Given that we are only sampling the position of the grippers and the use of VisCheck to define the goal region, we need to carefully define the distance and NearestNeighbour (Alg. 3) functions. Let $q_{xyz,rand}$ be the sampled gripper configuration. Let $R_{tree}$ be the set of all nodes $r_i = (q_{xyz,i}, B_i)$ in the tree. Then the distance between the sample and a node in the tree is the Euclidean distance betweeen gripper positions:

$$d(q_{xyz,rand}, r_i) = \|q_{xyz,rand} - q_{xyz,i}\| \ . \tag{4}$$

Let $R_{comp} = \{(q_{xyz}, B) \mid q_{xyz} = q_{xyz,goal}\} \setminus R_{goal}$ be the set of all RRT configurations whose gripper positions match the goal configuration, but whose virtual elastic bands are similar to the Blacklist. Then the distance between any configuration

**Algorithm 2** PlanPath($q_t, \mathcal{P}_t, B_t, \mathcal{T}, L_{max}$, Blacklist)

$\mathcal{T}_U \leftarrow$ UncoveredTargetPoints($\mathcal{P}_t, \mathcal{T}$)
$q_{xyz,goal} \leftarrow$ ClusterCenters($\mathcal{T}_U$)
$q_{xyz,goal} \leftarrow$ ProjectOutOfCollision($q_{xyz,goal}$)
$r_{start} \leftarrow (q_{xyz,t}, B_t)$
$R_{goal} \leftarrow \{(q_{xyz,goal}, B_{goal}) \mid \text{VisCheck}(B_{goal}, \text{Blacklist}) = 0\}$
$R_{nnbl} \leftarrow \emptyset$
Path $\leftarrow$ RRTPlan($r_{start}, R_{goal}, R_{nnbl}, L_{max}$)
**if** Path $\neq$ Failure **then**
    **return** ShortcutSmooth(Path)
**else**
    **return** Failure
**end if**

**Algorithm 3** NearestNeighbor($q_{rand}, q_{xyz,goal}, R_{tree}, R_{nnbl}$)

**if** $\|q_{rand} - q_{xyz,goal}\| = 0$ **and** $R_{tree} \setminus R_{nnbl} \neq \emptyset$ **then**
    $r_{near} \leftarrow \text{argmin}_{r \in R_{tree} \setminus R_{nnbl}} d(q_{rand}, r)$
    $R_{nnbl} \leftarrow R_{nnbl} \cup \{r_{near}\}$
**else**
    $r_{near} \leftarrow \text{argmin}_{r \in R_{tree}} d(q_{rand}, r)$
**end if**
**return** $(r_{near}, R_{nnbl})$

$r_{comp} \in R_{comp}$ and any $r_{goal} \in R_{goal}$ is zero. If $R_{tree} \cap R_{comp} \neq \emptyset$ then when the goal is sampled, the closest node in the tree cannot be extended towards the goal. To address this issue, we maintain a set of nodes that we have extended towards the goal $R_{nnbl}$. When $q_{xyz,goal}$ is selected as $q_{xyz,rand}$, we first consider any nodes $r \in R_{tree} \setminus R_{nnbl}$ which we have not yet extended towards the goal, returning the closest such node. If $R_{tree} \setminus R_{nnbl} = \emptyset$ then we have extended every node in the tree towards the goal already. In this case, we default to standard nearest neighbor behavior. While this possibility is remote it can happen, particularly near the start of planning when there are few nodes in the tree. If the random sample is not the goal, then we use standard nearest neighbour behavior, setting $r_{near}$ to the closest configuration in $R_{tree}$.

The combination of local control, deadlock prediction, and global planning are shown in the MainLoop function (Alg. 4). Because the virtual elastic band is an approximation we need to predict deadlock while executing the planned path. We use the same prediction method for path execution as for the local controller.

To set the maximum band length $L_{max}$ used by the global planner and the deadlock prediction algorithms, we calculate the geodesic distance between the grippers through the deformable object when it is relaxed and scale it by the task specified maximum stretching factor $\lambda$.

## 5 Experiments and Results

We now present four example tasks to demonstrate our algorithm, two with cloth, and two with rope. In the first and second tasks, two grippers manipulate the cloth so

**Table 1** Deadlock prediction parameters

| | | |
|---|---|---|
| Prediction Horizon | $N_p$ | 10 |
| Band Annealing Factor | $\alpha$ | 0.3 |
| History Window | $N_h$ | 100 |
| Error Improvement Threshold | $\beta_e$ | 1 |
| Configuration Distance Threshold | $\beta_m$ | 0.03 |

**Algorithm 4** MainLoop($\lambda, N_p$)

$D \leftarrow$ GeodesicDistanceBetweenGrippers($\mathcal{P}_{relaxed}$) //
$L_{max} \leftarrow \lambda D$; $t \leftarrow 0$ // Initialization
Blacklist $\leftarrow \emptyset$; Path $\leftarrow \emptyset$ //
$q_0 \leftarrow$ SenseRobotConfig(); $\mathcal{P}_0 \leftarrow$ SensePoints() //
**while** $\neg\Omega(\mathcal{P}_t)$ **do**
  $\mathcal{T} \leftarrow$ GetTargets($\mathcal{P}_t$)
  $B_t \leftarrow$ InitializeBand($\mathcal{P}_t$)
  **if** PredictDeadlock($q_t, \mathcal{P}_t, B_t$,Path, $\mathcal{T}, L_{max}, N_p$) **then**
    Blacklist $\leftarrow$ Blacklist $\cup \{B_t\}$ //
    Path $\leftarrow$ PlanPath($\mathcal{P}_t, B_t, \mathcal{T}, L_{max}$, Blacklist) //
    **if** Path = Failure **then** // Global planning
      **return** Failure //
    **end if** //
  **end if**
  **if** Path $\neq \emptyset$ **then**
    $\dot{q}_{cmd} \leftarrow$ FollowPath(Path)
    **if** PathFinished(Path) **then**
      Path $\leftarrow \emptyset$
    **end if**
  **else**
    $\dot{q}_{cmd} \leftarrow C(q_t, \mathcal{P}_t, \mathcal{T}, \rho, \Omega)$ // Local controller
  **end if**
  CommandConfiguration($q_t + \dot{q}_{cmd}$) //
  $q_{t+1} \leftarrow$ SenseRobotConfig(); $\mathcal{P}_{t+1} \leftarrow$ SensePoints() // Update
  $t \leftarrow t+1$ //
**end while**
**return** Success

that it covers a table. In the first task the cloth is obstructed by a pillar while in the second task the grippers must pass through a narrow passage before the table can be covered. The third and fourth scenarios require the robot to navigate a rope through three-dimensional maze before aligning the rope with a line traced on the floor (see Figure 1).[1]

All experiments were conducted in the open-source Bullet simulator [7], with additional wrapper code developed at UC Berkeley. The cloth is modeled as a triangle mesh using 1500 vertices with a total size of 0.3m $\times$ 0.5m. The rope is modeled as a series of small capsules linked together by springs. In the first rope experiment we use 39 capsules for a 0.78m long rope, and 47 capsules for a 0.94m rope in the last experiment. We emphasize that our method does not have access to the model of the deformable object or the simulation parameters. The simulator is used as a "black box" for testing. We set the maximum stretching factor $\lambda$ to 1.17 for the cloth and 1.1 for the rope. All tests are performed using an i7-7700K processor. We use the same deadlock prediction parameters for all tasks, shown in Table 1.

To smooth the path returned by the RRT, at each iteration we randomly select either a single gripper or both grippers and two configurations in the path. To smooth between the configurations we use the same forward-propagation method for the

[1] The video accompanying this paper (https://youtu.be/548ypyLOgPA) shows the task executions.

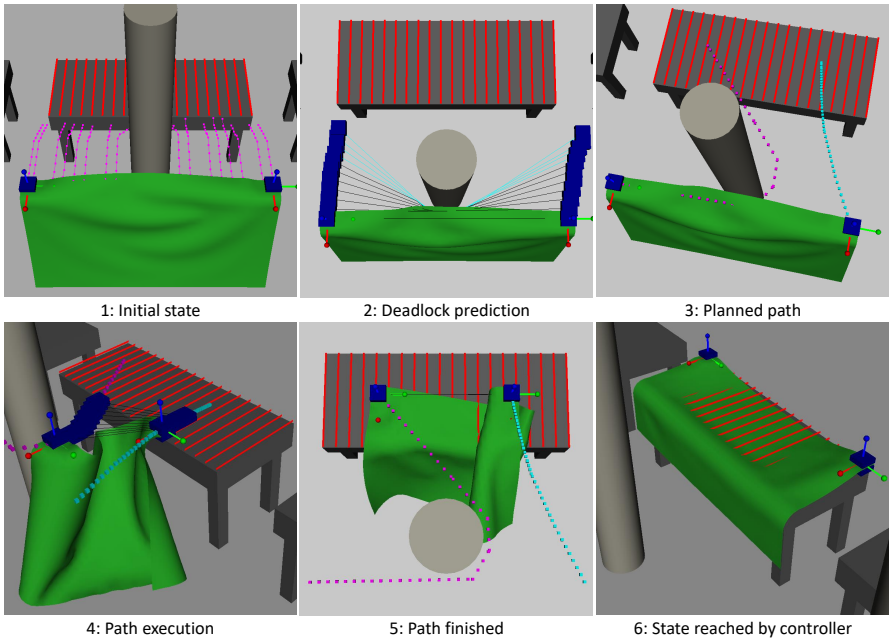


**Fig. 2** Sequence of snapshots showing the execution of the first experiment. The cloth is shown in green, the grippers are shown in blue, and the target points are shown as red lines. (1) The approximate integration of the potential fields from error reduction over $N_p$ timesteps, shown in magenta, pull the cloth to opposite sides of the pillar. (2) A sequence of *virtual elastic bands* between the grippers is shown in black and teal, indicating the predicted gripper configuration over the prediction horizon as the local controller follows the potential fields. The elastic band changes to teal as the predicted motion of the grippers moves the cloth into an infeasible configuration. (3 - 5) The resulting plan by the RRT, shown in magenta and teal, moves the system into a new neighbourhood. (6) Final system state when the task is finished by the local controller.

virtual elastic band as used in the planning process. If we have selected only one gripper for smoothing, we do not change the configuration of the second gripper during that smoothing iteration. We also forward-propagate the virtual elastic band to the end of the path to ensure that the band at the end of the smoothed path is disimilar from the blacklist. We perform 500 smoothing iterations for experiments 1, 2, and 4; and 1500 for experiment 3 due to the larger environment.

## *5.1 Single Pillar*

In the first example task, the objective is to spread the cloth across a table that is on the far side of a pillar (see Figure 2). We uniformly discretize the surface of the table to create the target points $\mathcal{T}$, with each discretized point creating a potential field that pulls the closest point on the deformable object towards the target. These target points

**Table 2** Local controller and deadlock prediction avg. computation time per iteration for each type of deformable object, averaged across all trials.

| | GetTargets() Time (s) | Predict Deadlock() Time (s) | Local Controller Time (s) |
|---|---|---|---|
| Cloth | 0.0108 | 0.0136 | 0.0056 |
| Rope | 0 | 0.0216 | 0.0029 |

**Table 3** Planning statistics for the first plan for each example task, averaged across 10 trials.

| | RRT Planning | | | | | Smoothing | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Samples | States | NN Time (s) | Validity Checking Time (s) | Total Time (s) | Iterations | Validity Checking Time (s) | Visibility Deformation Time (s) | Total Time (s) |
| Single Pillar | 369 | 2401 | $\sim 0.0$ | 2.9 | 3.0 | 500 | 2.4 | $\sim 0.0$ | 2.5 |
| Double Slit | 723 | 3204 | $\sim 0.0$ | 3.0 | 3.1 | 500 | 6.2 | $\sim 0.0$ | 6.2 |
| Rope Maze | 5099 | 9886 | 0.2 | 9.7 | 10.1 | 1500 | 12.6 | $\sim 0.0$ | 12.8 |
| Repeated Planning | 578 | 1626 | 0.1 | 2.9 | 3.0 | 500 | 4.7 | $\sim 0.0$ | 4.7 |

are set slightly above the surface to allow for collision margins within the simulator. A single point on the cloth can have multiple "pulls" or none. Task error $\rho$ is defined as the sum of the Dijkstra's distances from each target point to the closest point on the cloth. If a target point in $\mathcal{T}$ is within a small-enough threshold of their nearest neighbors in $\mathcal{P}$, then these points are considered "covered" and do not influence task error or any other calculation. Our results show that even though the global planner is only planning using the gripper positions and a virtual elastic band between them, it is able to find the correct neighbourhood for the local controller to complete the task. On average we are able to find and smooth a path in well under 10 seconds (Table 3), with the majority of the time spent on forward propagation of the virtual elastic band as part of the validity check for a potential movement of the grippers. In all 10 trials the global planner is only invoked once, with the local controller completing the task after the plan finishes. The average times for the local controller and deadlock prediction algorithms are shown in Table 2, averaged across all trials of all experiments.

## *5.2 Double Slit*

The second experiment uses the same setup as the first, with the only change being replacing the single pillar obstacle with a wide wall with two narrow slits (Figure 3). This adds a narrow passage problem and also demonstrates the utility of the progress detection filter. In this example the local controller is trying to move the deformable object straight forward, but with the wall in the way it is unable to make progress; the local controller cannot explicitly go around obstacles. This experiment shows comparable planning time, but it takes longer to smooth the resulting path (as expected given that the virtual elastic band forward propagation takes longer near obstacles). The local controller is again able to complete the task after invoking the planer a single time on all 10 trials.

## *5.3 Moving a Rope Through a Maze*

In the third task, the robot must navigate a rope through a three-dimensional maze before aligning the rope with a line traced on the floor (Figure 4). In this task, the correspondences between the target points $\mathcal{T}$ and the deformable object points $\mathcal{P}$ are fixed in advance, thus the GetTargets function does not have to do any work, as shown in Table 2. Task error $\rho$ is defined in the same way as in the first two

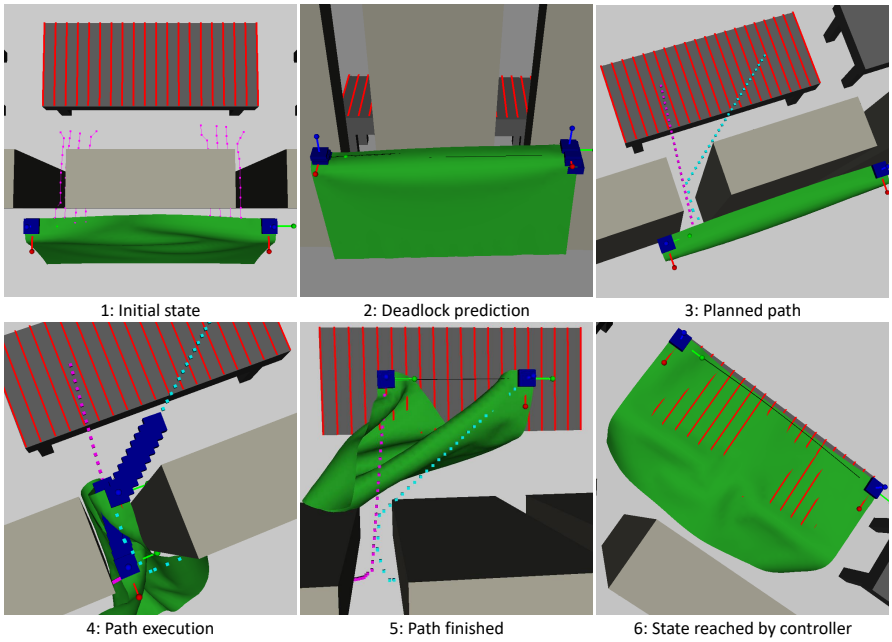


**Fig. 3** Sequence of snapshots showing the execution of the second experiment. We use the same colors as the previous experiment (Figure 2), but in this example instead of detecting future over-stretch in panel (2), we detect that the system is stuck in a bad local minimum and unable to make progress.

experiments. Again the planner is invoked a single time per trial, and planning time is comparable to the previous tasks, but we do spend more time smoothing for this example. This is a function of the size of the environment rather than any particular difference in the difficulty of performing the smoothing. This task has the most potential for a planned path to move the deformable object into a configuration from which the local controller cannot finish the task (by wrapping the rope around an obstacle near the goal), however our framework is able to address this as shown in the next example.

### *5.4 Repeated Planning*

The fourth task is a variant of the third, with the start point for the rope moved near the goal region on the top layer of the maze. For this experiment we reduce the size of the planning arena to only the goal area, and the immediate surroundings on the top layer (Figure 5). From this starting position, the planner is more likely to find the incorrect neighbourhood for the local controller on the first attempt. In 4 of the 10 trials, the planner was invoked twice, in 4 other trials it was invoked three times, and in one trial it was invoked four times. These additional planning and smoothing stages took on average an additional 40.9 seconds, but the task was completed successfully in all 10 trials.

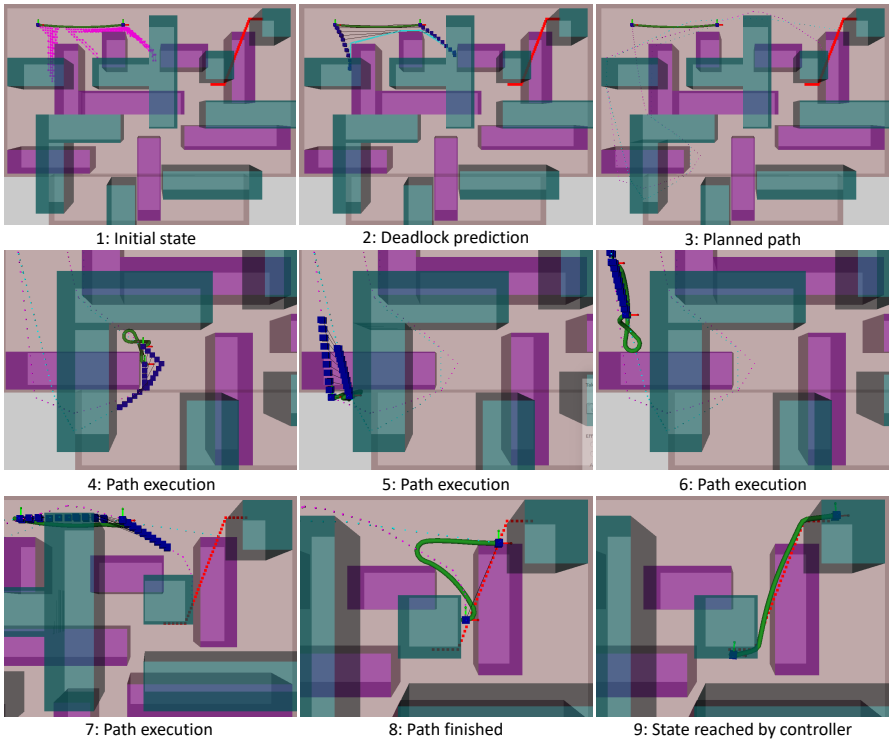


**Fig. 4** Sequence of snapshots showing the execution of the third experiment. The rope is shown in green starting in the top left corner, the grippers are shown in blue, and the target points are shown in red in the top right corner. The maze consists of top and bottom layers (green and purple, respectively). The rope starts in the bottom layer and must move to the target on the top layer through an opening (bottom left or bottom right).

## 6 Discussion and Conclusion

We have presented a method to interleave global planning and local control for deformable object manipulation that does not rely on high-fidelity modeling or simulation of the object. Our method combines techniques from topologically-based motion planning with a standard RRT implementation in order to generate gross motion of the deformable object. The purpose of this gross motion is not to achieve the task alone, but rather to move the object into a position from which the local controller is able to complete the task. This division of labour enables each component to focus on their strengths rather than attempt to solve the entire problem directly. As part of our framework, we introduced a novel deadlock prediction algorithm to determine when to use the local controller and when to use the global planner. Our experiments demonstrate that our framework is able to be applied to several interesting tasks for rope and cloth, including an adversarial case where we setup the planner to fail on the first attempt. In principle, these techniques can be used to extend our work beyond two grippers by using a virtual rubber band between every pair of grippers, however we have not experimented with such a task.

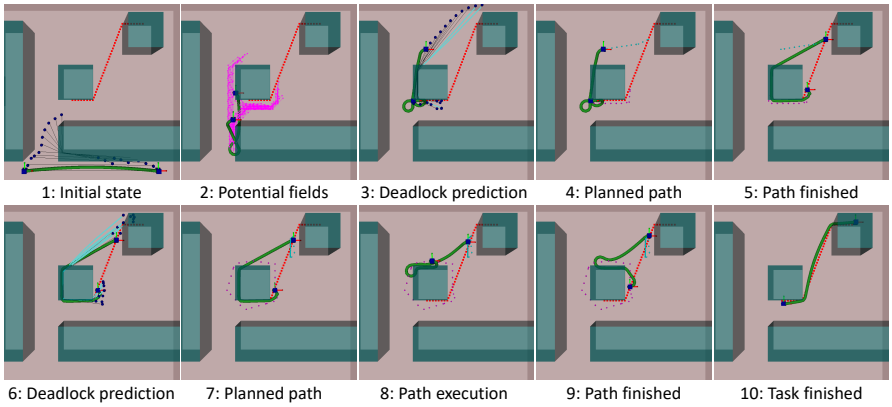


**Fig. 5** Sequence of snapshots for the fourth experiment. We use the same colors as the previous experiment (Figure 4), but in this example the local controller gets stuck twice, in panels 3 and 6. In panel 7 the global planner finds a new neighbourhood that is distinct from previously-tried neighbourhoods.

While we have included elements to handle the need to plan multiple times, there are several issues left unaddressed. In particular environments with "hooks" can cause problems due to our approximation methods; the virtual elastic band we use for constraint checking and planning assumes that there is no minimum length of the deformable object. This assumption means that our planner cannot detect cases where the excess material can get snagged on corners or hooks, preventing the motion plan from being executed. In addition we have no explicit method to avoid twisting or knot-tying behaviour. While shortcut smoothing can potentially mitigate the worst effects, this is not something that is within the scope of this work. Last, we cannot guarantee that we can achieve any given task in general; our sampling and nearest neighbour functions do not guarantee probabilistic completeness, and while our blacklisting method is designed to encourage exploration of the state space, it also has the potential to block regions of the state space from which the local controller can achieve the task. Despite these limitations we find that our framework is able to reliably perform complex tasks where neither planning nor control alone are sufficient.

In future work we plan to address these weaknesses, in particular the snagging and twisting limitations which are artifacts of our approximation methods. We also seek to extend our framework to a broader range of tasks, beyond coverage and point matching applications.

## References

1. Anshelevich, E., Owens, S., Lamiraux, F., Kavraki, L.: Deformable volumes in path planning applications. In: ICRA (2000)
2. Bai, Y., Yu, W., Liu, C.K.: Dexterous Manipulation of Cloth. Computer Graphics Forum **35**(2), 523–532 (2016)
3. Berenson, D.: Manipulation of deformable objects without modeling and simulating deformation. In: IROS (2013)
4. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Autonomous Robots **33**(3), 273–290 (2012)

5. Brass, P., Vigan, I., Xu, N.: Shortest path planning for a tethered robot. Computational Geometry **48**(9), 732–742 (2015)
6. Burchan Bayazit, O., Jyh-Ming Lien, Amato, N.: Probabilistic roadmap motion planning for deformable objects. In: ICRA (2002)
7. Coumans, E.: Bullet physics library. Open source: bulletphysics.org (2010)
8. Essahbi, N., Bouzgarrou, B.C., Gogu, G.: Soft Material Modeling for Robotic Manipulation. In: Applied Mechanics and Materials (2012)
9. Frank, B., Stachniss, C., Abdo, N., Burgard, W.: Efficient motion planning for manipulation robots in environments with deformable objects. In: IROS (2011)
10. Gayle, R., Lin, M., Manocha, D.: Constraint-Based Motion Planning of Deformable Robots. In: ICRA (2005)
11. Hirai, S., Wada, T.: Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model. Robotica **18**(1), 3–11 (2000)
12. Huang, S.H., Pan, J., Mulcaire, G., Abbeel, P.: Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects. In: IROS (2015)
13. Jaillet, L., Siméon, T.: Path deformation roadmaps: Compact graphs with useful cycles for motion planning. The International Journal of Robotics Research **27**(11-12), 1175–1188 (2008)
14. Jiménez, P.: Survey on model-based manipulation planning of deformable objects. Robotics and Computer-Integrated Manufacturing **28**(2), 154–163 (2012)
15. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12**(4), 566–580 (1996)
16. Khalil, F., Payeur, P.: Dexterous robotic manipulation of deformable objects with multi-sensory feedback – a review. In: In-Teh (ed.) Robot Manipulators, Trends and Development, chap. 28, pp. 587–621 (2010)
17. Lamiraux, F., Kavraki, L.E.: Planning Paths for Elastic Objects under Manipulation Constraints. The International Journal of Robotics Research **20**(3), 188–208 (2001)
18. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006)
19. McConachie, D., Berenson, D.: Bandit-Based Model Selection for Deformable Object Manipulation. WAFR (2016)
20. Moll, M., Kavraki, L.E.: Path Planning for Deformable Linear Objects. IEEE Transactions on Robotics **22**(4), 625–636 (2006)
21. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation, vol. 29. CRC Press (1994)
22. Navarro-Alarcon, D., Liu, Y.h., Romero, J.G., Li, P.: On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. The International Journal of Robotics Research **33**(11), 1462–1480 (2014)
23. Quinlan, S.: Real-time modification of collision-free paths. Ph.D. thesis, Department of Computer Science, Stanford University (1994)
24. Rodriguez, S., Amato, N.: An obstacle-based rapidly-exploring random tree. In: ICRA (2006)
25. Roussel, O., Borum, A., Taïx, M., Bretl, T.: Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations. In: ICRA (2015)
26. Saha, M., Isto, P., Latombe, J.C.: Motion planning for robotic manipulation of deformable linear objects. In: Proc. International Symposium On Experimental Robotics (ISER) (2006)
27. Schulman, J., Ho, J., Lee, C., Abbeel, P.: Learning from demonstrations through the use of non-rigid registration. In: Springer Tracts in Advanced Robotics, vol. 114, pp. 339–354. Springer International Publishing (2016)
28. Smolen, J., Patriciu, A.: Deformation Planning for Robotic Soft Tissue Manipulation. In: 2009 Second International Conferences on Advances in Computer-Human Interactions, pp. 199–204 (2009)
29. Soonkyum Kim, Likhachev, M.: Path planning for a tethered robot using Multi-Heuristic A* with topology-based heuristics. In: IROS (2015)
30. Wada, T., Hirai, S., Kawarnura, S., Karniji, N.: Robust manipulation of deformable objects by a simple PID feedback. In: ICRA (2001)