Manipulation of Deformable Objects Without Modeling and Simulating Deformation

Dmitry Berenson

***Abstract*— We present a method to manipulate deformable objects that does not require modeling and simulating deformation. Our method is based on the concept of diminishing rigidity, which we use to quickly compute an approximation to the Jacobian of the deformable object. This Jacobian is used to drive the points within the deformable object towards a set of targets. However, this Jacobian alone is insufficient to avoid stretching the object beyond its allowed length and to avoid gripper collision with obstacles. Thus a key part of our approach is incorporating techniques to avoid collision and excessive stretching. Our experiments show how to perform several interesting tasks for one and two-dimensional deformable objects using our method. They also show how the method can be applied to collaborative tasks, where the robot and a user simultaneously manipulate the deformable object. Our experiments are conducted in simulation but we emphasize that our method does not have access to the model of the deformable object used by the simulator, although we assume we are able to sense the geometry of the object. While our method is local, we find that it is quite versatile in the range of tasks it can perform, especially since it has no knowledge of the model of the deformable object.**

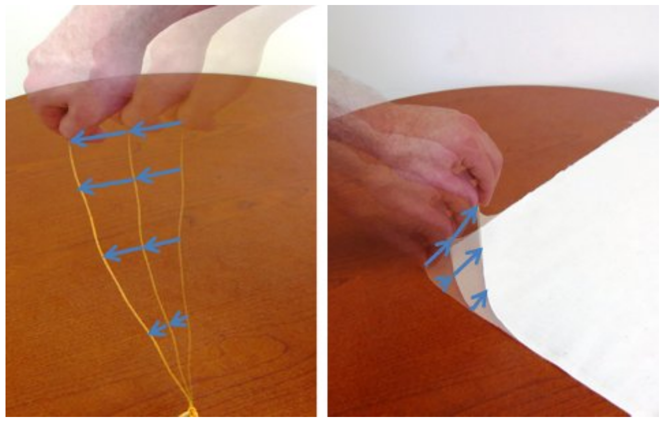

Fig. 1. Two examples showing the concept of diminishing rigidity. Left: Moving twine which is fixed at one end. Right: Picking up the corner of a piece of cloth. In both examples parts of the object near gripped point move similarly to the gripper (i.e. as if they were rigidly attached to the gripper), while parts that are farther away from the gripper have smaller displacements.

I. INTRODUCTION

Many real-world manipulation tasks involve deformable objects. From folding a bed sheet in the home to tying suture in the operating room, the ability to manipulate deformable objects is an important capability for assistive and autonomous robots to poses.

The primary challenge of manipulating deformable objects is that they are very difficult to model and simulate. Unlike rigid objects whose dynamics are well-understood, the motion of a deformable object depends on a large and complex set of parameters that define its stiffness, friction, and volume preservation. As a result, a great deal of research has focused on how to model these kinds of objects. It is possible to model known deformable objects in known environments (such as an industrial or manufacturing setting [1]). However, online modeling of deformable objects in new environments is currently an open problem, though some progress has been made in probing deformable objects to acquire model parameters [2], [3].

Even if it were possible to obtain a perfect model of the deformable object, simulating that model presents its own challenges. Many methods exist for deformable object simulation, including mass-spring model simulation [4] [5] and the more general Finite Element Method (FEM) [6]. Simulation methods for mesh-less models also exist [7]. Specialized domain-specific simulators have also been developed; for instance for organ deformation during needle insertion [8]. FEM simulation is generally regarded to be the most accurate, although it is highly sensitive to the discretization of the deformable object and, for fine discretizations, is very time-consuming.

Given the difficulties of deformable object modeling and simulation, we seek to explore the practicality of manipulating deformable objects without explicitly modeling and simulating them. Our hypothesis is that model-free deformable manipulation like this can be accomplished by exploiting a property we call *diminishing rigidity* – i.e. that the effect of gripper motion along the deformable object diminishes as the distance from the gripper increases (see Figure 1 for motivating examples). This property is based on the assumption that the combination of gravity, friction, and stretching of the deformable object dissipate the force applied at the gripper. The more distant a point is from the gripper, the more the above forces can counteract the force at the gripper. Of course this property does not hold for all deformable objects in all situations. For instance, imagine a cloth that is draping off a table; a small pull of the tablecloth could cause it to slip off the table completely. However, our experiments show that assuming diminishing rigidity does indeed yield useful behavior for several non-trivial tasks.

In this paper we address the problem of manipulating deformable objects locally, i.e. given a desired displacement of the points comprising the object we compute a motion of the grippers that achieves that displacement as closely as

Dmitry Berenson is with the Department of Computer Science and the Robotics Engineering Program, Worcester Polytechnic Institute (WPI), 100 Institute Road, Worcester, MA, 01609 USA dberenson@cs.wpi.edu

possible. To do this we use the diminishing rigidity property to quickly compute an approximation to the Jacobian of the deformable object. This Jacobian is used to iteratively move the points comprising the deformable object toward a set of targets. However, this Jacobian alone is insufficient to avoid stretching the object beyond its allowed length and to avoid gripper collisions with obstacles. Thus a key part of our approach is incorporating techniques to avoid collision and excessive stretching.

In our experiments we show how to encode several interesting tasks for one and two-dimensional deformable objects (i.e. rope and cloth), as instances of this kind of problem. Surprisingly, our straightforward method can also be applied to collaborative tasks, where the robot and a user manipulate the deformable object together. Our experiments are conducted using the open-source Bullet simulator [9], however we emphasize that the simulator is used as a "black box" to test the algorithm, which does not require any knowledge of the underlying model. However, we do assume that we are able to sense the shape of the deformable object (although the sensing can be noisy). This experimental setup is intended to mimic the information available in the physical world as closely as possible.

II. RELATED WORK

Robotic manipulation of deformable objects has been studied in many contexts ranging from surgery to industrial manipulation (see [1] for an extensive survey).

Motion planning for manipulation of deformable objects is an active area of research. Saha et al. [10] present a Probabilistic Roadmap that plans for knot-tying tasks with rope. Rodriguez et al. [11] study motion planning in fully-deformable simulation environments. Their method, based on Rapidly-exploring Random Trees, 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. [12] presented a method that pre-computes deformation simulations in a given environment to enable fast multi-query planning. We see the local method presented in this paper as potentially useful for motion planning, as it can generate useful primitive actions for consideration by the planner. Also, the above methods rely on physical simulation of the deformable object, while our method does not.

The method presented in this paper is most closely related to work in visual servoing for deformable objects. Smolen and Patriciu [13] formulate an iterative Jacobian-based method to align interest points on an object with targets. They model the object using the mesh-less Reproducing Kernel Particle Method and use this model to compute the Jacobian. Hirai and Wada [14] propose an iterative visual-servoing controller which aligns interest points on the deformable object to targets. Their control law is based on modeling the deformable object as a lattice of interconnected springs. Wada et al. [15] use a similar spring model to formulate a PID controller that aligns interest points to targets. Our work differs from those above in that we do not rely on an explicit model to represent the deformable object. Also, unlike our method, the above methods do not consider obstacle avoidance or compensation for excessive stretching, which are key to practical deformable object manipulation.

Our iterative Jacobian-based method is derived from the operational space control of robot manipulators [16]. Similar methods are used to solve inverse-kinematics problems for high-DOF robots [17]. We also make use of the null-space of the Jacobian to allow movements that do not interfere with constraints (in our case obstacle avoidance), similar to Sentis and Khatib [18].

III. PROBLEM STATEMENT

Let the configuration of a deformable object be defined by the position of the points within the object. Let $\mathcal{P} \subset \mathbb{R}^3$ be a discrete set of points of size P that approximates the deformable object. We assume that $\mathcal{P}$ can be sensed, though it may not be sensed precisely due to sensor error. We allow obstacles to be present in the environment of the deformable object as well as standard gravity. We assume that the obstacle geometry can be sensed.

Let the robot have G grippers which grasp the deformable object. We assume that each grasp is a rigid pinch grasp, i.e. the points on the deformable object that are pinched by the plates of the gripper cannot move with respect to the gripper. We assume that each gripper can move independently in all six degrees of freedom in $SE(3)$. Let a configuration of the G grippers be $q \in \mathbb{Q}$. Let the function $F : \mathbb{Q} \to \mathbb{R}^{3P}$ map the gripper configuration q to the position of the points in the deformable object. We assume that the grippers move slowly enough that we can treat the system as quasi-static.

In this paper we assume that the grippers are free-floating, i.e. they are unattached to robot arms. Robot arm kinematics are well-studied in the literature and the Jacobians computed for free-floating grippers presented here can easily be computed for grippers attached to robot arms instead, however this work is not within the scope of this paper.

The problem we address in this paper is how to move the G grippers such that the points in $\mathcal{P}$ align as closely as possible with some set of target points $\mathcal{T} \subset \mathbb{R}^3$ while avoiding gripper collision and excessive stretching of the deformable object. Let ρ be a function that computes the alignment error between $\mathcal{P}$ and $\mathcal{T}$. The method we present is local, i.e. at each time t it chooses an incremental movement $\dot{q}$ which reduces the alignment error as much as possible at time $t+1$:

$$\min_{\dot{q}_t} \rho(\mathcal{T}, \mathcal{P}_{t+1}) \tag{1}$$

where $\mathcal{P}_{t+1} = F(q_t + \dot{q}_t)$ is the result of performing the motion $\dot{q}_t$ at time t. q_t must also be feasible, i.e. it should not bring the grippers into collision with obstacles and should not cause the object to stretch excessively. We will show that several practical tasks can be represented as instances of this problem.

IV. A LOCAL METHOD FOR DEFORMABLE OBJECT MANIPULATION

Solving the problem in Equation 1 exactly in the general case is impractical for two 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. Thus we present an approximate solution to this problem which can be computed quickly.

Our solution to Equation 1 is to construct an approximate linearization of the F function by computing an approximation to the Jacobian of F:

$$\begin{aligned}\mathcal{P} &= F(q) \\ \frac{\partial \mathcal{P}}{\partial t} &= \frac{\partial F(q)}{\partial q}\frac{\partial q}{\partial t} \\ \dot{\mathcal{P}} &= J(q)\dot{q} \qquad (2)\end{aligned}$$

The Jacobian $J(q)$ allows us to drive $\mathcal{P}$ toward $\mathcal{T}$, however the Jacobian alone is insufficient to avoid stretching the object beyond its allowed length and to avoid collision with obstacles. Thus we integrate methods to mitigate these issues.

A. *Computing an approximate* $J(q)$

The typical approach to computing an approximation to the Jacobian at a given configuration is to perturb each degree of freedom of q independently by a small amount, simulate the system for some amount of time, and collect the corresponding changes in state in the Jacobian matrix. For instance, this type of approach was applied for deformable objects in [13]. In our problem domain, such an approach would require a fast and accurate simulation of the deformable object in the given environment, which we do not assume to have. This method also scales poorly with the number of grippers. In testing on our simulator (described in the Experiments section), we found that this method was highly unreliable in computing an approximate Jacobian and was very sensitive to the size of the perturbation applied (despite being quite time-consuming).

The key observation that enables us to compute an approximate Jacobian efficiently is that deformable objects tend to be *locally rigid* near points that are grasped by the gripper. Since we assume the object is grasped tightly, points grasped by the gripper clearly move rigidly with respect to the gripper. However, we also observe that points nearby to the points that are grasped move "almost-rigidly," and points farther from the gripped points move "less rigidly." We can use this observation to compute an approximation to $J(q)$ which we call the *diminishing rigidity Jacobian* $\tilde{J}(q)$.

The first step to computing $\tilde{J}(q)$ is to establish baseline geodesic distances between the points in $\mathcal{P}$. We thus assume that we have knowledge of the object in its "natural" non-deformed state. For instance, for a rope this would be the rope stretched out in a straight line at its maximum extent. For a cloth, it would be the cloth laid flat, etc. In this natural state, for two given points $p_i, p_j \in \mathcal{P}$, we can compute the geodesic distance between them $d(p_i, p_j)$ by finding the shortest path between p_i and p_j that is contained in the object[1]. We then construct a matrix D of size $P \times P$, where $D_{i,j} = d(p_i, p_j)$. Note that this computation is done offline so it does not add to the online computational load of our method.

We now show how to compute $\tilde{J}(q)$ by first computing the translation and rotation Jacobians for each point in $\mathcal{P}$ and each gripper. Let the set of points grasped by the gripper having index g be $\mathcal{C}_g \subseteq \mathcal{P}$. The closest point in $\mathcal{C}_g$ to the ith point in $\mathcal{P}$ is

$$c(i,g) = \operatorname*{arg\,min}_{p \in \mathcal{C}_g} D_{i,\mathcal{P}(p)}, \qquad (3)$$

where $\mathcal{P}(p)$ is the index of point p in $\mathcal{P}$.

We define $w(i,g)$ to be the "rigidity" of the ith point with respect to gripper g. Many functions could be used to compute rigidity as long as they obey the following properties:

1) $w(i,g)$ ranges between 0 and 1
2) $w(i,g) = 1$ if the ith point is grasped by gripper g
3) $w(i,g) > w(j,g)$ if $D_{i,\mathcal{P}(c(i,g))} < D_{j,\mathcal{P}(c(j,g))}$

The first property is necessary because of the way rigidity is used in the Jacobian below. The second property follows directly from the assumption that grasped points move rigidly with the gripper. The third property encodes the concept of diminishing rigidity as described in the Introduction.

Through experimentation, we found that we obtained good results when rigidity decreased exponentially with distance at a rate of k, which we tune experimentally:

$$w(i,g) = e^{-k(D_{i,\mathcal{P}(c(i,g))})}. \qquad (4)$$

Note that this function meets the required properties above.

This rigidity function will be used to weigh the components of $\tilde{J}(q)$. The translation component of $\tilde{J}(q)$ for the point in $\mathcal{P}$ with index i and gripper with index g is

$$\tilde{J}_{trans}(q,i,g) = \mathbf{I}_{3\times3}. \qquad (5)$$

Let $\mathbf{R}^g$ be the 3×3 rotation matrix, and v^g be the 3×1 vector of translation of the gripper having index g in the world frame. Let $r = (c(i,g) - v^g)$. The rotation component of $\tilde{J}(q)$ for the ith point in $\mathcal{P}$ and gripper with index g is then

$$\tilde{J}_{rot}(q,i,g) = [\mathbf{R}^g[1] \times r,\ \mathbf{R}^g[2] \times r,\ \mathbf{R}^g[3] \times r], \qquad (6)$$

where $\mathbf{R}^g[n]$ represents the nth column of $\mathbf{R}^g$. Combining $\tilde{J}_{trans}(q,i,g)$ and $\tilde{J}_{rot}(q,i,g)$ and scaling by the rigidity, we get

$$\tilde{J}(q,i,g) = w(i,g)\left[\tilde{J}_{trans}(q,i,g),\ \tilde{J}_{rot}(q,i,g)\right], \qquad (7)$$

which is size 3×6.

Finally, combining the Jacobians for all points in $\mathcal{P}$ and all G grippers, we get

[1]For convex objects, this is simply the Euclidean distance between p_i and p_j.

$$\tilde{J}(q) = \begin{bmatrix} \tilde{J}(q,1,1) & \tilde{J}(q,1,2) & \cdots & \tilde{J}(q,1,G) \\ \tilde{J}(q,2,1) & \ddots & & \\ \vdots & & & \\ \tilde{J}(q,P,1) & & & \end{bmatrix}, \quad (8)$$

which is size $3P \times 6G$.

We can then apply the Moore-Penrose pseudo-inverse[2] of $\tilde{J}(q)$ to compute a movement of the grippers $\dot{q}$ which produces a desired $\dot{\mathcal{P}}$:

$$\dot{q} = \tilde{J}(q)^{+}\dot{\mathcal{P}}. \quad (9)$$

B. Correcting for Excessive Stretching

While applying $\dot{q}$ will indeed drive the points in $\mathcal{P}$ toward a given target, Equation 9 does not take into account constraints on allowable stretching of the deformable object. For instance, imagine a rope which is fixed at one end and held by a gripper at the other. If using Equation 9 to drive the rope toward a target which is farther from the fixed point than the length of the rope, the rope could be torn. Such cases become difficult to detect in the presence of obstacles, where, for instance, a rope may wind around an obstacle.

To mitigate this problem, we first compute the Euclidean distance between every pair of points in $\mathcal{P}$ at each time step and store it in the $P \times P$ matrix E. We can then compare E to D, which contains the maximum allowed lengths between every pair of points, to detect if two points are near their maximum distance apart. We can then attempt to move such points closer. Let $\dot{\mathcal{P}}_s$ be desired displacements of the nodes to correct for excessive stretching. We use the $\mathrm{StretchingCorrection}$ function (Algorithm 1) to compute $\dot{\mathcal{P}}_s$ based on a user-defined threshold λ. We can then include $\dot{\mathcal{P}}_s$ in Equation 9:

$$\dot{q} = \tilde{J}(q)^{+}(\dot{\mathcal{P}} + \dot{\mathcal{P}}_s) \quad (10)$$

C. Obstacle Avoidance

While contact between the deformable objects and obstacles is expected (and in some cases desired), the grippers are assumed to be rigid and a practical implementation requires that they not collide with obstacles. One approach to local obstacle avoidance is to repel each gripper from the nearest object while projecting the $\dot{q}$ resulting from Equation 10 into the null space of the avoidance motion:

$$\dot{q}_g = J^{+}_{p^g}\dot{x}_{p^g} + (\mathbf{I} - J^{+}_{p^g}J_{p^g})\dot{q}_g \quad (11)$$

where J_{p^g} is the Jacobian for the closest point on gripper g to the nearest obstacle and $\dot{x}_{p^g}$ is a displacement for that point that repels it from the nearest obstacle. The $\dot{q}_g$ on the right-hand side of the equation is the movement computed for the gripper with index g in Equation 10. Let $\mathcal{O}$ be the set of obstacles. J_{p^g} and $\dot{x}_{p^g}$ are computed using the

Algorithm 1: StretchingCorrection(E, D, λ, $\mathcal{P}$)

```
1  $\dot{\mathcal{P}}_s \leftarrow \mathbf{0}_{3P\times 1}$;
2  $\Delta \leftarrow E - D$;
3  for $i \in \{1,2,...,P\}$ do
4      for $j \in \{i, i+1, ..., P\}$ do
5          if $\Delta_{i,j} - \lambda < 0$ then
6              $v \leftarrow \Delta_{i,j}(\mathcal{P}_j - \mathcal{P}_i)$;
7              $\dot{\mathcal{P}}_{s_i} \leftarrow \dot{\mathcal{P}}_{s_i} + \frac{1}{2}v$;
8              $\dot{\mathcal{P}}_{s_j} \leftarrow \dot{\mathcal{P}}_{s_j} - \frac{1}{2}v$;
9          end
10     end
11 end
12 return $\dot{\mathcal{P}}_s$;
```

Algorithm 2: ObstacleAvoidance(g, $\mathcal{O}$)

```
1  $d_g \leftarrow \infty$;
2  for $o \in \{1,2,...,|\mathcal{O}|\}$ do
3      $p^g, p^o \leftarrow \mathrm{ClosestPoints}(g, o)$;
4      $v \leftarrow p^g - p^o$;
5      if $\|v\| < d_g$ then
6          $d_g \leftarrow \|v\|$;
7          $\dot{x}_{p^g} \leftarrow \frac{v}{\|v\|}$;
8          $J_{p^g} \leftarrow \mathrm{GripperPointJacobian}(g, p^g)$;
9      end
10 end
11 return $\{J_{p^g}, \dot{x}_{p^g}, d_g\}$;
```

$\mathrm{ObstacleAvoidance}$ function[3] (Algorithm 2). $(\mathbf{I} - J^{+}_{p^g}J_{p^g})$ is the null space of J_{p^g}.

While Equation 11 will move the grippers away from collision while allowing null-space motion to align $\mathcal{P}$ and $\mathcal{T}$, the repulsion is active regardless of the distance from the obstacle. Initially, we only used Equation 11 when a gripper came within some threshold of an obstacle and otherwise use Equation 10. However this lead to jitter as the system switched between the two equations around the threshold. Instead, we developed a way to merge the two equations so that the transition is smooth. We first compute a severity measure γ_g, ranging from 0 (far from collision) to 1 (in contact):

$$\gamma_g = e^{-\beta d_g} \quad (12)$$

where d_g is the distance from the closest obstacle to the closest point on the gripper with index g and β is a user-defined positive parameter. We then balance the movements computed by Equations 10 and 11 using γ_g:

$$\dot{q}_g = \gamma_g(J^{+}_{p^g}\dot{x}_{p^g} + (\mathbf{I} - J^{+}_{p^g}J_{p^g})\dot{q}_g) + (1-\gamma_g)\dot{q}_g \quad (13)$$

In this way there is a smooth transition between the two desired movements. As d_g approaches 0, the collision avoid-

[2] $A^{+} = (A^TA)^{-1}A^T$ for any matrix A

[3] The ObstacleAvoidance function requires computing the closest point on a gripper to an obstacle, which can be implemented using freely-available collision detectors (we use the Bullet collision detector).

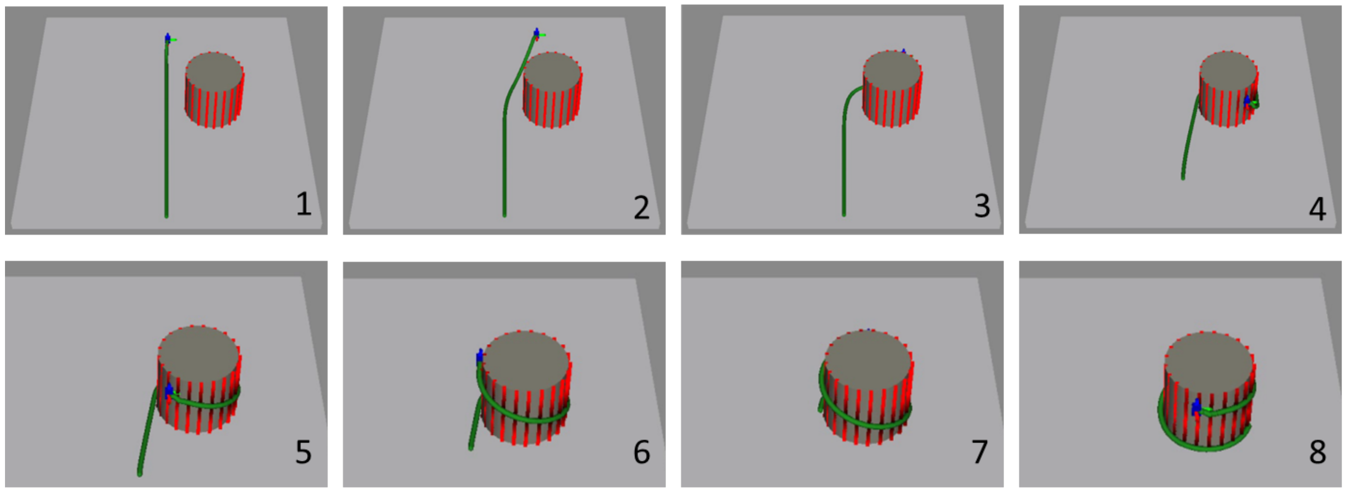

Fig. 2. Sequence of snapshots showing the execution of the first experiment. The rope is shown in green and the gripper is shown in blue. The points on the cylinder range from red (distant from the closest point of the rope) to black (close to the closest point of the rope).

Algorithm 3: MainLoop(D, $\mathcal{O}$, α, β, λ)

```
1  $t \leftarrow 0$;
2  **while** *true* **do**
3      $\mathcal{P}_t \leftarrow \text{SensePoints}()$;
4      $q_t \leftarrow \text{SenseGripperConfiguration}()$;
5      $\mathcal{T} \leftarrow \text{GetTargets}(\mathcal{P}_t)$;
6      $\dot{\mathcal{P}} \leftarrow \text{DesiredDisplacement}(\mathcal{P}_t, \mathcal{T})$;
7      $\tilde{J} \leftarrow \text{LocalRigityJacobian}(q_t, D)$;
8      $E \leftarrow \text{EuclidianDistanceMatrix}(\mathcal{P}_t)$;
9      $\dot{\mathcal{P}}_s \leftarrow \text{StretchingCorrection}(E, D, \lambda, \mathcal{P}_t)$;
10     $\dot{q} \leftarrow \tilde{J}^{+}(\dot{\mathcal{P}} + \dot{\mathcal{P}}_s)$;
11     **for** $g \in \{1, 2, ..., G\}$ **do**
12         $J_{p^g}, \dot{x}_{p^g}, d_g \leftarrow \text{ObstacleAvoidance}(g, \mathcal{O})$;
13         $\gamma_g \leftarrow e^{-\beta d_g}$;
14         $\dot{q}_g \leftarrow \gamma_g(J_{p^g}^{+}\dot{x}_{p^g} + (\mathbf{I} - J_{p^g}^{+}J_{p^g})\dot{q}_g) + (1-\gamma_g)\dot{q}_g$;
15     **end**
16     **if** $||\dot{q}|| > \alpha$ **then**
17         $\dot{q} \leftarrow \alpha\frac{\dot{q}}{||\dot{q}||}$;
18     **end**
19     $\text{MoveToConfiguration}(q_t - \dot{q})$;
20     $t \leftarrow t + 1$;
21 **end**
```

ance term dominates the other term completely, thus guaranteeing obstacle avoidance.

The combination of obstacle avoidance, stretching correction, and the diminishing rigidity Jacobian are shown in the MainLoop function (Algorithm 3). This is the function used in our experiments.

V. EXPERIMENTS

We now present three example tasks where the above method is effective, show how to encode those tasks in the DesiredDisplacement and GetTargets functions, and discuss experimental results. The first task shows how our method can be applied to a rope, with the goal of winding the rope around a cylinder in the environment. The second and third tasks show the method applied to cloth. In the second task, two grippers manipulate the cloth so that it covers a table. In the third task, a user collaboratively folds the cloth with the assistance of autonomous grippers. The video accompanying this paper shows the task executions.

In our experiments we use a uniform discretization to generate the points on the object, however task-specific discretizations can also be used. In general, the more points a part of the object has, the more the alignment of that part to its target will influence the motion of the grippers.

All experiments were conducted in the open-source Bullet simulator [9], with additional wrapper code developed at UC Berkeley. The sizes of objects and parameters of the simulator were tuned to produce visually-realistic behavior. The rope is modeled as a series of 50 small capsules linked together by springs and is 25m long. The cloth is modeled as a triangle mesh size 10m $\times$ 10m. 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" to test our method. However, we do assume that we are able to sense the shape of the deformable object (although the sensing can be noisy). This experimental setup is intended to mimic the information available in the physical world as closely as possible.

A. *Winding a rope around a cylinder*

In the first example task, a single gripper holds a rope that is laying on a table. The task is to wind the rope around a cylinder which is also on the table (see Figure 2). We encode this task in the DesiredDisplacement as follows: First, we discretize the surface of the cylinder (except for the caps). These points are the $\mathcal{T}$ returned by the GetTargets function. The alignment error $\rho(\mathcal{P}, \mathcal{T})$ is then the sum of the distances between every point in $\mathcal{T}$ to the closest point in $\mathcal{P}$ in meters. The desired displacement for a point $p \in \mathcal{P}$ is then the sum of the vectors from p to all points in $\mathcal{T}$ for which p is the nearest point. Since the rope cannot occupy the same physical space as the cylinder, we disregard points in $\mathcal{T}$ if they are within a small-enough threshold of their nearest neighbors in $\mathcal{P}$.

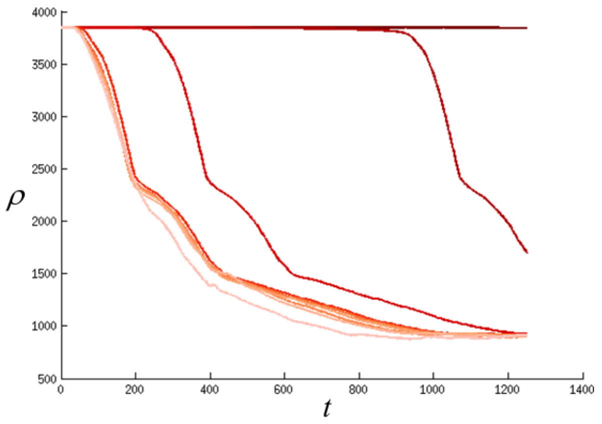

Fig. 3. Alignment error ρ vs. iteration number for the rope-winding task with k varying from 0.2 (lightest) to 2.0 (darkest) in steps of 0.2.

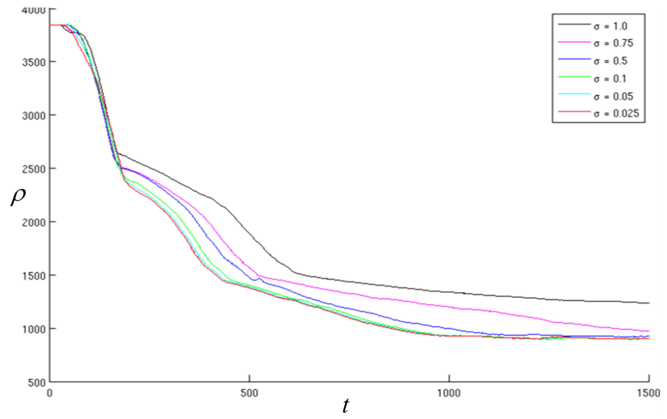

Fig. 4. Alignment error ρ vs. iterations of the main loop for the rope-winding task with varying amounts of noise on sensing $\mathcal{P}$. σ is the standard deviation of the zero-mean Gaussian noise added to $\mathcal{P}$.

These points are considered "covered" and do not influence the desired displacement (although their alignment error is still included in ρ).

The parameter values used for this experiment (established by trial and error) were $\lambda = 0.1$ and $\beta = 10$. We found that the rotation component of $\tilde{J}$ performed poorly for the rope, thus it was not used in this scenario.

To tune the rigidity parameter k we tested a range of values in this scenario (see Figure 3). Under-estimating the rigidity (i.e. a high k) resulted in the rope moving very slowly toward the cylinder in the beginning of the task. This is because the parts of the rope being "pulled" toward the cylinder by the $\mathrm{DesiredDisplacement}$ function are in the middle of the rope (these are the points that are closest to the cylinder). When the rigidity is low the magnitude of $\tilde{J}$ is small for points in the middle of the rope because they are distant from the gripper, thus the resulting displacement is small. Our method converged to the minimum error the fastest for the lowest value of $k = 0.2$, however we chose $k = 0.5$ for subsequent experiments because $k = 0.2$ seemed to over-fit to the task in this scenario and $k = 0.5$ was more robust for general manipulation with the rope, especially with noise.

Results for this experiment with varying levels of noise in the perception of the points of the rope are shown in Figure 4. To simulate sensor uncertainty, we perturbed the position of the points of the object reported by the simulator with zero-mean Gaussian random noise (generated using the Box-Muller method). Since the cylinder is much larger than the rope, it is impossible to cover all the points on the surface of the cylinder even if the entire rope is wound around it, hence the high steady-state alignment error shown in Figures 3 and 4. Once the rope was wound around the cylinder, the gripper exhibited a periodic motion of continuing to circle around the cylinder while maintaining a near-constant error. Our results in Figure 4 show that the method degraded gracefully with increasing noise in this scenario.

B. Spreading a cloth across a table

The second scenario we consider is spreading a cloth across a table (see Figure 5). In this scenario two grippers hold the rectangular cloth at two corners and the

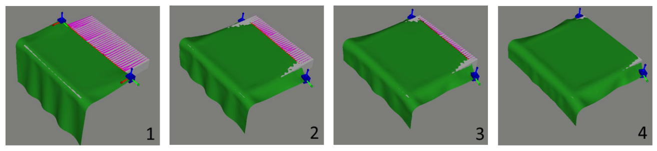

Fig. 5. Sequence of snapshots showing the execution of the second experiment. The cloth is shown in green and the grippers are shown in blue. The purple lines on the table show the distance from points on the table surface to the closest points on the cloth. The slight penetration of the cloth into the table is due to the inaccuracy of the collision-handling used by the simulator.

task is to cover the top of the table with the cloth. The $\mathrm{DesiredDisplacement}$ and $\mathrm{GetTargets}$ functions are defined similarly to the previous example, except that the target points used are a discretization of the top of the table. We use both the translation and rotation parts of $\tilde{J}$ in this example. We found that our method generally over-estimated the effect of rotation in this scenario, thus we scaled the rotation component of $\tilde{J}$ by 50 to compensate. The parameters used for this experiment were $\lambda = 0.1$, $k = 0.7$, and $\beta = 100$.

As in the previous example, we perturbed the position of the points of the object reported by the simulator with zero-mean Gaussian random noise. Results for this experiment with varying levels of noise are shown in Figure 7. Even when the cloth fully covers the table it is practically impossible to achieve an error of 0 because the points on the cloth $\mathcal{P}$ will never perfectly match the points of the table surface $\mathcal{T}$. Unlike the previous example, our method was more sensitive to noise in this scenario, with higher levels of noise causing divergence. We believe this is due to errors in the handling of collisions between the cloth and table in the simulator. Higher noise sometimes caused the grippers to mistakenly pull the cloth taught across the table, which caused inter-penetration in the simulator. We also observed that higher noise would cause the system to move into undesirable local minima where the cloth became bunched or folded. Because our method is local, it was not able to escape from these minima. For lower noise, the method successfully covered the table with the cloth.

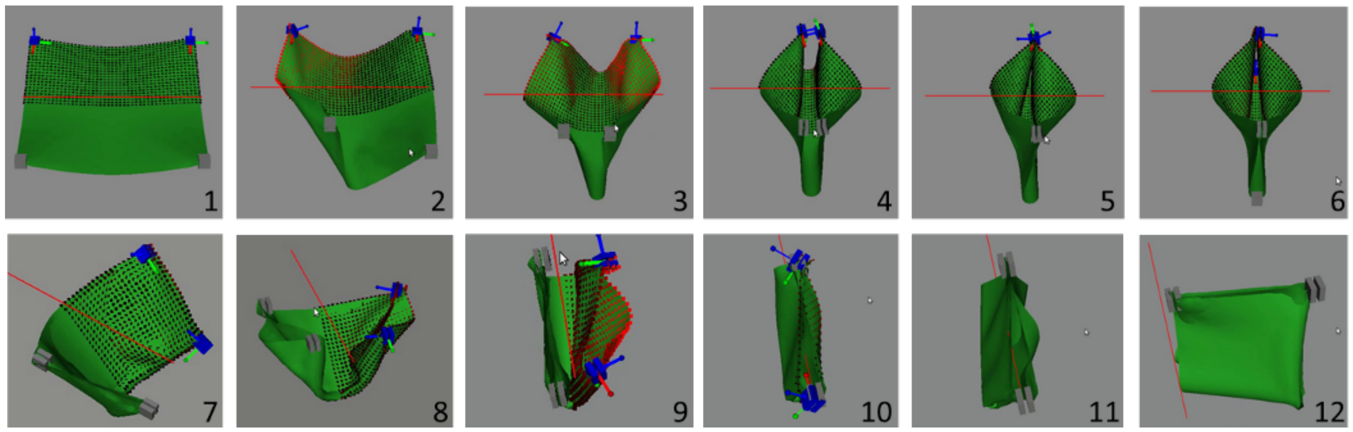

Fig. 6. Sequence of snapshots showing the execution of the third experiment. The autonomous grippers are blue, and the grippers controlled by the user are gray. The red line marks the plane of symmetry used by the DesiredDisplacement function for this experiment. The points on the autonomous grippers' side of the cloth $\mathcal{P}_{auto}$ range from red (distant from the corresponding point in $\mathcal{T}$) to black (close to the corresponding point in $\mathcal{T}$).

C. Collaborative Cloth Folding

In this experiment, a user uses two grippers to collaborate with two autonomous grippers to fold a cloth twice. This task is inspired by the common laundry task of folding a bed-sheet (see Figure 6). The user controls his or her grippers using a mouse for translation and keyboard for rotation.

We found that intuitive collaboration between the user and the autonomous grippers could be accomplished through *symmetry preservation*. The DesiredDisplacement function is defined so that the autonomous grippers attempt to match their side of the cloth to the user's side of the cloth. To encode this behavior, we define a plane of symmetry halfway between the user's grippers and the autonomous grippers at their initial configuration (note that this plane is static, it does not move with the cloth). To encode symmetry preservation, before the task begins we reflect the points of the cloth on the user's side $\mathcal{P}_{user}$ across the plane of symmetry, and find their corresponding points on the autonomous side $\mathcal{P}_{auto}$.

Online, at each iteration, we first reflect the current sensed $\mathcal{P}_{user}$ across the plane of symmetry to produce $\mathcal{T}$ (this is computed in the GetTargets function). The alignment error $\rho(\mathcal{P}, \mathcal{T})$ is then the sum of the distances between every point in $\mathcal{T}$ and its corresponding point in $\mathcal{P}_{auto}$. The DesiredDisplacement function then returns the set of vectors $\mathcal{T} - \mathcal{P}_{auto}$, which are the desired displacements of the points in $\mathcal{P}_{auto}$. The desired displacement of points in $\mathcal{P}_{user}$ is set to 0, so the user's side is not intentionally disrupted.

We use both the translation and rotation parts of $\tilde{J}$ in this example. Unlike the previous experiment, no rotation scaling was needed. The parameter values were the same as above.

Re-grasping operations are required to complete the cloth-folding task. However, grasping deformable objects is a complex problem [19] and not within the scope of this paper. Instead we place the grippers at manually-selected locations on the cloth and close them. To accomplish the last part of the fold, the autonomous grippers must hand-off the cloth to the user's grippers. Doing this while avoiding collision is not possible with the grasps and gripper models we use, so collision avoidance between the grippers was not used here.

Figure 6 shows the execution of the collaborative cloth folding task. In frames 1-5 the user brings the gray grippers into alignment for the first re-grasp while the autonomous grippers preserve symmetry for their side of the cloth. As seen in frame 5, the error between $\mathcal{P}_{auto}$ and $\mathcal{T}$ is low (the points on autonomous gripper side are black). Frame 6 shows the first re-grasp, where one of the user's grippers re-grasps the top of the cloth while the other grasps the bottom-most part of the cloth. The autonomous grippers perform the same re-grasp on their side. In frames 7-9 the user brings the cloth into alignment for the second re-grasp while the autonomous grippers preserve symmetry. The user then brings the gray grippers to the plane of symmetry so that they overlap with the autonomous grippers (frame 10). The user's grippers re-grasp the cloth in frame 11 and the autonomous grippers are removed. The user displays the folded cloth in frame 12.

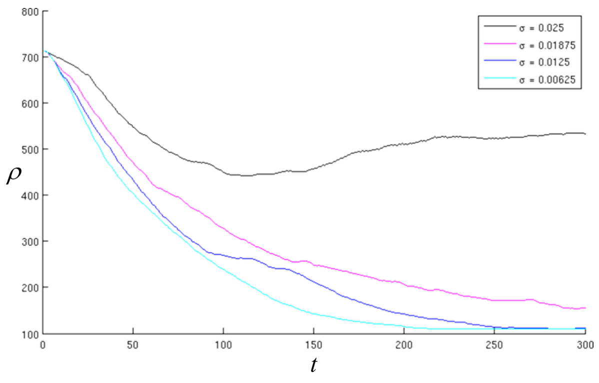

Fig. 7. Alignment error ρ vs iteration number for the cloth-spreading task with varying amounts of noise on sensing $\mathcal{P}$. σ is the standard deviation of the zero-mean Gaussian noise added to the $\mathcal{P}$ given by the simulator.

D. Compensating for Excessive Stretching

While stretching compensation was active in the above experiments, it is difficult to gauge its effects without deliberately attempting to tear the deformable object. Thus we also conducted an experiment to show the system's ability to compensate for excessive stretching of a cloth. Again, the user controls two grippers while the other two grippers are autonomous. The same symmetry preservation described above is active in this example.

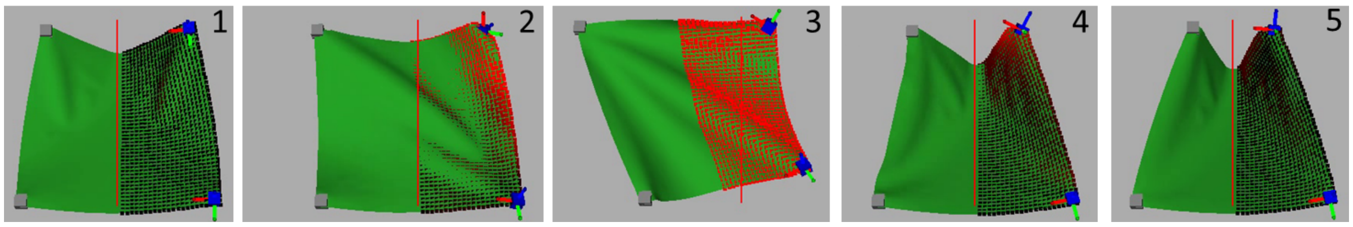

Fig. 8. Sequence of snapshots showing the ability of our method to correct for excessive stretching.

Figure 8 shows the execution of the stretching experiment. The cloth starts in a state where symmetry has been established (frame 1). The user's topmost gripper pulls the cloth away from the center (frames 2-3). The symmetry preservation is overridden by the stretching correction and the topmost autonomous gripper moves toward the center to avoid excessive stretching of the cloth (frames 2-3). This causes the points in $\mathcal{P}_{auto}$ to move farther from their corresponding points in $\mathcal{T}$, which is why the points become more red. When the user's gripper moves closer to the center (frame 4), the effect of stretching compensation goes to 0 and the symmetry preservation is re-established (frame 5).

VI. DISCUSSION AND CONCLUSIONS

We have presented a method to manipulate deformable objects that does not rely on modeling or simulation of the object. Our local method combines an approximate Jacobian based on diminishing rigidity with compensation for excessive stretching and collision-avoidance. This method is able to drive points on the deformable object toward targets while simultaneously enforcing constraints. Our experiments show how to perform several interesting tasks for rope and cloth using our method. They also show how the method can be applied directly to collaborative tasks, where the robot and a human manipulate the deformable object together.

While we demonstrated that our approach can be applied to manipulating rope and cloth in our experiments, it is not well-suited for all scenarios involving deformation. Heterogeneous deformable objects would be particularly problematic for our approach because the diminishing rigidity may vary widely depending on where the object is grasped. Our approach also assumes that the grippers move quasi-statically, and thus it cannot be applied to situations where highly dynamic motion is necessary to accomplish a task. Despite these limitations, we find that our method is quite versatile in the range of tasks it can perform, especially since it has no knowledge of the model of the deformable object.

In future work we seek to explore methods for automatically-tuning the k parameter online (perhaps using methods similar to [20]), instead of tuning it manually. We believe automatic tuning will allow the Jacobian to adapt quickly to new situations and will diminish the effects of local minima. We also seek to extend our method to three-dimensional deformable objects.

ACKNOWLEDGMENTS

We gratefully acknowledge John Schulman and Jonathan Ho for their help with Bullet. We also thank Pieter Abbeel and Ken Goldberg for their support and helpful discussions.

REFERENCES

[1] F. Khalil and P. Payeur, "Dexterous robotic manipulation of deformable objects with multi-sensory feedback – a review," in *Robot Manipulators, Trends and Development*, In-Teh, Ed., 2010, ch. 28, pp. 587–621.

[2] J. Lang, D. Pai, and R. Woodham, "Acquisition of elastic models for interactive simulation," *The International Journal of Robotics Research (IJRR)*, vol. 21, no. 8, pp. 713–733, 2002.

[3] A. Cretu, P. Payeur, and E. Petriu, "Neural network mapping and clustering of elastic behavior from tactile and range imaging for virtualized reality applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 9, pp. 1918–1928, 2008.

[4] M. Desbrun, P. Schröder, and A. Barr, "Interactive animation of structured deformable objects," in *Graphics Interface*, 1999.

[5] Z. Pezzementi, D. Ursu, S. Misra, and A. M. Okamura, "Modeling Realistic Tool-Tissue Interactions with Haptic Feedback: A Learning-based Method," in *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Mar. 2008, pp. 209–215.

[6] P. Kaufmann, S. Martin, M. Botsch, and M. Gross, "Flexible simulation of deformable models using discontinuous Galerkin FEM," in *SIGGRAPH*, 2008.

[7] F. Faure, B. Gilles, G. Bousquet, and D. K. Pai, "Sparse meshless models of complex deformable solids," *SIGGRAPH*, vol. 1, no. 212, p. 1, 2011.

[8] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. Hauser, K. Goldberg, J. Shewchuk, and J. O'Brien, "Interactive simulation of surgical needle insertion and steering," in *SIGGRAPH*, 2009.

[9] E. Coumans, "Bullet physics library," *Open source: bulletphysics.org*, 2006.

[10] M. Saha, P. Isto, and J.-C. Latombe, "Motion planning for robotic manipulation of deformable linear objects," in *Proc. International Symposium On Experimental Robotics (ISER)*, 2006.

[11] S. Rodriguez, J.-M. Lien, and N. Amato, "Planning motion in completely deformable environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[12] B. Frank, C. Stachniss, N. Abdo, and W. Burgard, "Efficient motion planning for manipulation robots in environments with deformable objects," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 2180–2185.

[13] J. Smolen and A. Patriciu, "Deformation Planning for Robotic Soft Tissue Manipulation," in *2009 Second International Conferences on Advances in Computer-Human Interactions*, Feb. 2009, pp. 199–204.

[14] S. Hirai and T. Wada, "Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model," *Robotica*, vol. 18, no. 1, pp. 3–11, Jan. 2000.

[15] T. Wada, S. Hirai, S. Kawarnura, and N. Karniji, "Robust manipulation of deformable objects by a simple PID feedback," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[16] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Transactions on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[17] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*, 2nd ed. Springer, 2000, pp. 96–100.

[18] S. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, pp. 505–518, December 2005.

[19] K. Gopalakrishnan and K. Goldberg, "D-space and deform closure grasps of deformable parts," *International Journal of Robotics Research (IJRR)*, vol. 24, November 2005.

[20] D. Navarro-Alarcon, Y. Liu, J. Romero, and P. Li, "Visually Servoed Deformation Control by Robot Manipulators," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.