Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints

Yixuan Wang[1], Dale M[c]Conachie[1], and Dmitry Berenson[1]

***Abstract*— In order to manipulate a deformable object, such as rope or cloth, in unstructured environments, robots need a way to estimate its current shape. However, tracking the shape of a deformable object can be challenging because of the object's high flexibility, (self-)occlusion, and interaction with obstacles. Building a high-fidelity physics simulation to aid in tracking is difficult for novel environments. Instead we focus on tracking the object based on RGBD images and geometric motion estimates and obstacles. Our key contributions over previous work in this vein are: 1) A better way to handle severe occlusion by using a motion model to regularize the tracking estimate; and 2) The formulation of *convex* geometric constraints, which allow us to prevent self-intersection and penetration into known obstacles via a post-processing step. These contributions allow us to outperform previous methods by a large margin in terms of accuracy in scenarios with severe occlusion and obstacles.**

I. Introduction

Tracking the shape of a deformable object is a long-standing problem that has been studied for applications in computer graphics [1], [2], surgery [3], [4], computer vision [5], [6], and robotics [7], [8], [9]. However, tracking deformable objects in the presence of severe occlusion and obstacles remains a difficult open problem. The key challenge is to maintain an estimate of the shape that conforms with physical constraints, i.e. that the object's motion conforms to a reasonable motion model and that the object cannot move through obstacles or through itself. Occlusion makes enforcing these kinds of constraints especially difficult.

Previous work on this problem has explored using a physics simulator to inform the prediction estimate [10], but such methods assume a simulation environment for the given scene (with appropriate friction, stiffness, etc. parameters) can be easily constructed. Even if this were the case, such methods are quite sensitive to occlusion. More recently, CDCPD [7] showed that such models were not necessary for accurate tracking, instead using geometric methods to infer the shape of occluded parts of the object. However, CDCPD has two key limitations: 1) It has no motion model to constrain how the object moves between frames and, as a result, large erroneous changes in state are possible due to occlusion; and 2) There is no way to enforce geometric constraints for self-intersection and obstacle penetration.

This paper builds on CDCPD to address these key deficiencies. Specifically, our method, CDCPD2, makes three novel additions: 1) We incorporate a user-defined motion model into the regularization process to bias the tracker toward realistic object motion; 2) We formulate convex constraints that prevent edge-crossing and incorporate them into a post-processing step; and 3) We formulate convex obstacle-penetration constraints and include them in post-processing. While we use a similar expectation-maximization procedure as CDCPD, we emphasize that the above additions are highly non-trivial and, more importantly, that they allow us to track deformable objects in much more realistic scenarios, where occlusion and clutter are unavoidable.

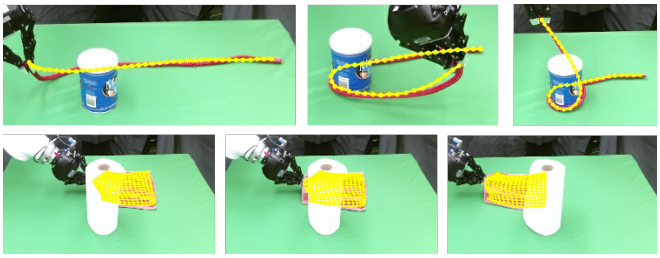

Fig. 1: Examples of tracking results for rope winding around a cylinder (top) and cloth behind an obstacle (bottom). The 3D tracking estimate (yellow) is overlaid on the corresponding image.

Our experiments compare our method to CDCPD and [10]. Our simulation results (where we can obtain the ground-truth state of the object) show that CDCPD2 is able to estimate the state of the object much better than the previous methods under severe occlusion. Our real-world results show a clear qualitative improvement in the estimate of the deformable object shape in several challenging scenarios. Our code is available open-source[2].

II. Related Work

Deformable object tracking has been studied in a variety of fields. Due to space constraints, we focus on only the most relevant methods to our work. In computer vision, [5] explores methods to reconstruct non-rigid scenes in real-time. However, it does not track a soft object model explicitly, as new nodes can be added and removed. For self-occlusion, [6] introduces a method to handle self-occlusion based on feature matching between between a template and input images. However, the deformable objects we consider don't have obvious features to use for matching.

Deformable object tracking has also been used for image-guided surgery. However, these approaches typically rely on textural features [3] or a high-fidelity simulation of the deformable object to inform tracking [4]. In contrast, our method does not require either of these and thus it is applicable to a broader range of objects and novel scenarios.

In robotics, [11] introduces the method of representing a textured wire as a NURBS spline model and manipulates the

[1]Yixuan Wang, Dale M[c]Conachie, and Dmitry Berenson are with the University of Michigan, Ann Arbor, MI, USA. {yixuanwa, dmcconac, dmitryb}@umich.edu

[2]https://github.com/UM-ARM-Lab/cdcpd/tree/CDCPD2

wire based on this model. However, it requires the wires to have a specific color pattern. [12] focuses on tracking with obstacles and occlusion. However, it is based on the known Finite Element Method (FEM) model of the specific object, which can be difficult to construct. [13] focuses on grasping cloth, where the tracking of the cloth is based on corner detection. [14] tracks and manipulates rope but is specialized to knot tying applications, whereas we seek to track both rope and cloth-like objects. [15], [16] follows a similar approach, using hand-engineered features for perception.

Unlike much of the above work, which uses hand-engineered features for perception, several recent methods use point registration based on Gaussian Mixture Model Expectation-Maximization (GMM-EM) and Coherent Point Drift (CPD) (e.g. [17]). Tang *et al.* proposes to incorporate physics simulation into the point registration process [8], [9], [10]. This method is effective for knotting and folding, but it performs poorly when the object is significantly occluded (by itself or by obstacles) and it requires setting up a simulation with many friction parameters and material properties that are difficult to determine for a given deformable object. Chi and Berenson [7] introduce a method to track rope and cloth while allowing for occlusions of the object without relying on a physics simulator. However, their method does not address obstacle interactions and self-intersection. We compare to both [7] and [10] in our experiments.

III. Problem Statement

Let $\langle \mathcal{P}^t, \mathcal{E} \rangle$ to be the configuration of the deformable object at time t. $\mathcal{P}^t \in \mathbb{R}^{3M}$ is a set of M points, and every element $p_i \in \mathcal{P}^t$ is a point with coordinate (x, y, z). $\mathcal{E} \in \mathbb{I}^{2E}$ is a set of E edges, and every element $e_i \in \mathcal{E}$ has two indices of $\mathcal{P}^t$ which form an edge. We assume that we know the poses of the robot's G gripper(s). The gripper(s) configuration at time t is denoted as $q^t \in SE(3)^G$ with velocity $\dot{q}^t$. Each element $\dot{q}_g \in \dot{q}^t$ can be written as $[v_g^T \ \omega_g^T]^T \in \mathfrak{se}(3)$, i.e. the concatenation of the translational and angular velocities. We denote indices of $\mathcal{P}^t$ grasped by G grippers at time t as $\mathcal{C}^t \in \mathbb{I}^{dG}$, where $c_i \in \mathcal{C}^t$ means d indices grasped by the ith gripper. In addition, we define ρ_{ij} as the geodesic distance distance between points p_i and p_j on the surface of the deformable object.

The inputs to the tracking algorithm are a sequence of RGBD images $\mathcal{I}^t$, q^t (optional), $\dot{q}^t$ (optional), $\mathcal{C}^t$ (optional), and an initial connectivity model $\langle \mathcal{P}^0, \mathcal{E} \rangle$. The optional may be used by some motion models, but it depends on the model. As with previous work ([10], [7]), we assume a point-cloud is generated from the RGBD image and the object is segmented out of the pointcloud. In this work, all points not belonging to the deformable object are assumed to be obstacles. We denote deformable object masks as $\mathcal{M}^t$, with the same size as RGBD images $\mathcal{I}^t$, and masked deformable objects' corresponding point clouds are denoted as $\mathcal{D}^t \subset \mathbb{R}^{3N}$, a set of N points, and every element $d_i \in \mathcal{D}^t$ is a point with coordinate (x, y, z). We assume that the obstacle points can be used to infer the shape of the obstacles (e.g. using shape-completion [18], [19], model registration [20], or simply creating a mesh from the points), and thus we obtain a mesh representation of obstacles $\mathcal{O}$.

Algorithm 1: CDCPD2

Input : $\mathcal{P}^0, \mathcal{E}, \mathcal{O}, \mathcal{C}^t_{\text{idx}}, \mathcal{I}^t, \mathcal{M}^t, q^t, \dot{q}^t, \mathcal{C}^t, t = 1, 2, ...$
Output: $\mathcal{P}^t, t = 1, 2, ...$
Compute G using Eq. 8;
Compute L using [21];
for $t = 0, 1, ...$ **do**
  Retrieve the masked point cloud $\mathcal{D}^{t+1}$ from $\mathcal{I}^{t+1}$;
  $\mathcal{P}^{\text{GMM},t+1} =$
   GMM-EM$(\mathcal{P}^t, \mathcal{D}^{t+1}, L, G, q^{t+1}, \dot{q}^{t+1}, \mathcal{C}^{t+1}, \mathcal{O})$;
  Optimize $\mathcal{P}^{t+1}$ using Eq. 24;

Tracking a deformable object can then be regarded as a point registration problem of estimating $\mathcal{P}^{t+1}$ by aligning $\mathcal{P}^t$ to $\mathcal{D}^{t+1}$, with known obstacles $\mathcal{O}$ and (optionally) gripper information $\langle q^{t+1}, \dot{q}^{t+1}, \mathcal{C}^{t+1} \rangle$. Denoting the true configuration of the deformable object at time t as $\mathcal{P}^{*t+1}$, our goal is to output the $\mathcal{P}^{t+1}$ that minimize $\|\mathcal{P}^{*t+1} - \mathcal{P}^{t+1}\|_2$. In addition, we wish to obtain an estimate with the same topology (in the sense of the over- and under- crossings considered in knot theory [14]) as the ground truth. This problem is difficult because the tracking method must compensate for (self-)occlusion, prevent self-intersection, and enforce the geometric constraints arising from obstacles.

IV. Method

As a foundation for our method, we build on Constrained Deformable Coherent Point Drift (CDCPD) [7]. At its core, CDCPD uses a Gaussian Mixture Model (GMM) Expectation-Maximization (EM) process to find the configuration of the deformable object $\mathcal{P}^{t+1}$ that best explains the observation $\mathcal{D}^{t+1}$ given the previous configuration $\mathcal{P}^t$ and the robot configuration q^{t+1} (Sec. IV-A). In addition, CDCPD uses Coherent Point Drift (Sec. IV-B.1) and Locally Linear Embedding (Sec. IV-B.2) to encourage the result to be physically plausible, however this is insufficient for some types of severe occlusion. In this paper, our first contribution is to introduce an additional novel regularization term based on a geometric model-based prediction (Sec. IV-B.3); this enables us to use the motion of the robot $(q^t \to q^{t+1})$ to help infer the motion of the deformable object $(\mathcal{P}^t \to \mathcal{P}^{t+1})$.

While these regularization terms encourage physically-plausible results, they do not exploit constraints that we know about the system. CDCPD enforces constraints based on the position of the robot grippers and the maximum size of the deformable object (Sec. IV-C.1). However, CDCPD does not account for self-intersection and obstacles. Thus our second contribution is to introduce two additional constraints: a self-intersection constraint (Sec. IV-C.2) that ensures that the tracked deformable object does not pass through itself, and an obstacle interaction constraint (Sec. IV-C.3) that ensures that the tacked object is never inside an obstacle. The overall method is show in Alg. 1.

A. Deformable Object Tracking as Point Set Registration

Assuming relatively small changes in object shape between two consecutive frames, it is reasonable to regard

the tracking problem as a point registration problem. In other words, we will align $\mathcal{P}^t$ to $\mathcal{D}^{t+1}$ to get $\mathcal{P}^{t+1}$. For simplicity, in this section, we will write $\mathcal{P}^t$ as $\mathcal{P}^{\text{prev}}$, $\mathcal{P}^{t+1}$ as $\mathcal{P}$, $\mathcal{P}^{\text{GMM},t+1}$ as $\mathcal{P}^{\text{GMM}}$, and $\mathcal{D}^{t+1}$ as $\mathcal{D}$. Following the formulation in [22], [23], [7], the problem will initially be formulated as a naive GMM problem.

We assume every point $p_i \in \mathcal{P}^{\text{GMM}}$ is the center of a Gaussian distribution with the same isotropic variance σ^2. The sum of M Gaussians is a Gaussian Mixture Model (GMM). In addition to the Gaussian terms, we add a uniform term with weight $w \in (0, 1)$ to the mixture. Thus

$$\mathsf{P}(m) = \begin{cases} \frac{1-w}{M}, & m = 1, \ldots, M \\ w, & m = M+1 \end{cases} \tag{1}$$

and

$$\mathsf{P}(d_n|m) = \begin{cases} \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \exp\left(-\frac{\|d_n - p_m\|^2}{2\sigma^2}\right), & m = 1, \ldots, M \\ \frac{1}{N}, & m = M+1 \end{cases} \tag{2}$$

We then assume that points in $\mathcal{D}$ were generated by drawing samples from the mixture:

$$\mathsf{P}(\mathcal{D}) = \prod_{n=1}^{N} \mathsf{P}(d_n) = \prod_{n=1}^{N} \sum_{m=1}^{M+1} \mathsf{P}(m)\mathsf{P}(d_n|m) \ . \tag{3}$$

Then our goal is to find the configuration of the deformable object that best explains the observation; i.e.

$$\underset{\mathcal{P}}{\operatorname{argmax}} \log \mathsf{P}(\mathcal{D}) \ . \tag{4}$$

The EM algorithm can be used to solve this problem, as shown in [22]. In the E-step we keep the Gaussian distributions fixed and calculate the expectation that d_n is generated by the m'th Gaussian distribution

$$\mathsf{P}(m|d_n) = \begin{cases} \frac{1}{\eta} \exp\left(-\frac{\|d_n - p_m\|^2}{2\sigma^2}\right), & m = 1, \ldots, M \\ \frac{1}{\eta} \frac{w}{(1-w)} \frac{(2\pi\sigma^2)^{\frac{3}{2}} M}{N}, & m = M+1 \end{cases} \tag{5}$$

where $\eta > 0$ is set so that $\sum_{m=1}^{M+1} \mathsf{P}(m|d_n) = 1$. Then in the M-step, following the algorithm described in [22], [23], we update the position of the Gaussian centroids $\mathcal{P}$ and variance σ by minimizing

$$Q(\mathcal{P}, \sigma^2) = \sum_{n=1}^{N} \sum_{m=1}^{M+1} \mathsf{P}(m|d_n) \frac{\|d_n - p_m\|^2}{2\sigma^2} + \frac{3}{2} N_{\mathsf{P}} \log \sigma^2 \tag{6}$$

where N_{P} is the sum of $\mathsf{P}(m|d_n)$, i.e. $\sum_{n,m=1}^{N,M} \mathsf{P}(m|d_n)$. Then we iterate E-step and M-step until convergence.

1) Visual Information Exploited for Occlusion: The base GMM-EM formulation assumes a uniform prior for each Gaussian centroid (see Eq. 1); CDCPD replaces this assumption based on visibility information. If a particular point on the deformable object is occluded, then that point is unlikely to generate any samples in $\mathcal{D}$. This information is encoded in the posterior probability matrix $X \in \mathbb{R}^{M \times N}$ (see Eq. (9) in [7]).

Algorithm 2: GMM-EM($\mathcal{P}^{\text{prev}}, \mathcal{D}, L, G, q, \dot{q}, \mathcal{C}, \mathcal{O}$)

Input : $\mathcal{P}^{\text{prev}}, \mathcal{D}, L, G, q, \dot{q}, \mathcal{C}, \mathcal{O}$
Output: $\mathcal{P}^{\text{GMM}}$
Compute p_{vis} using Eq. (7) in [7];
$\sigma^2_{\text{prev}} = 0$;
$\sigma^2 = \text{Var}(\mathcal{P}^{\text{prev}})$;
$\sigma^2_{\text{diff}} = \epsilon + 1$;
$W = 0$;
$i = 0$;
while $\sigma^2_{diff} > \epsilon$ *and* $i <$ *max_iter* **do**
  Compute X using Eq. (9) in [7];
  Compute $\mathcal{P}^{\text{pred}}$ using motion model;
  Solve W using Eq. 15;
  $\sigma^2_{\text{prev}} = \sigma^2$;
  Compute σ^2 using Eq. 16;
  $\sigma^2_{\text{diff}} = \text{abs}(\sigma^2 - \sigma^2_{\text{prev}})$;
  $i++$;
return $\mathcal{P}^{GMM} = \mathcal{P}^{prev} + GW$

B. GMM-EM Regularization

1) Coherent Point Drift Regularization: While GMM-EM forms a reasonable basis for a probabilistic interpretation of an observation; it does not account for dependencies between each Gaussian centroid. In our domain, there is physical structure to these centroids based on the deformable object; in particular two points on the deformable object that have a short geodesic distance between them are likely to move coherently. CPD encodes this structure by restricting the motion of the deformable object to be of the form

$$\mathcal{P}^{\text{GMM}} = \mathcal{P}^{\text{prev}} + GW \tag{7}$$

where G is a Gaussian kernel matrix with elements

$$G_{ij} = \exp\left(-\rho^2_{ij}/(2\beta^2)\right) \tag{8}$$

and $W \in \mathbb{R}^{M \times 3}$ is a weight matrix. We can then regularize this weight matrix to enforce motion coherence during the the M-step:

$$E_{CPD} = \text{Tr}\left(W^T G W\right) \ . \tag{9}$$

Note that we deviate from CDCPD by using a static value for G rather than recalculating it at every timestep.

2) Locally Linear Embedding Regularization: While CPD is able to encourage motion coherence during consecutive frames, it does not account for excessive change over time. For example in the extreme case, two points can drift arbitrarily far apart given enough time. To address this problem CDCPD uses a regularization term proposed in [23]. This term is based on Locally Linear Embedding (LLE) [21], which represents each point as a linear combination of its k nearest neighbours, and then penalizes deviation from that representation. We obtain linear weights L by minimizing the following cost function:

$$J(L) = \sum_{m=1}^{M} \|p^0_m - \sum_{i \in K_m} L_{mi} p^0_i\|^2 \tag{10}$$

where K_m is a set of indices for the k nearest neighbors of p^0_m, and L is a $M \times M$ adjacency matrix where L_{ij}

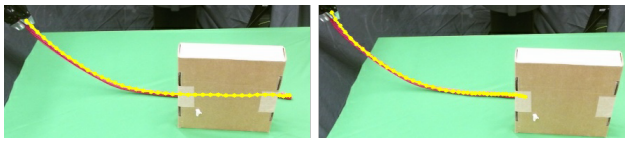

Fig. 2: CPCPD producing a large change in the state when the end of the rope becomes occluded. Left: before. Right: after.

represent a edge between p_i^0 and p_j^0 with their corresponding linear weight if $j \in K_i$ and 0 otherwise. We then define a regularization term that penalize the deviation from the original local linear relationship:

$$E_{LLE}(W) = \sum_{m=1}^{M} \|p_m^{\text{GMM}} - \sum_{i \in K_m} L_{mi} p_i^{\text{GMM}}\|^2 \tag{11}$$

where p_i^{GMM} are the points from the CPD transformation (Eq. 7).

3) Geometric Prediction-based Regularization: While CDCPD is able to handle some occlusion cases, others like those shown in Fig. 2 cause the algorithm to lose track of one side of the deformable object, shrinking the tracking to only the visible points. Though these two frames are close in time, the tracking result changes dramatically when the trailing end of the rope is occluded. Without any visual points to "pin" one side of the rope on the left side of the box, the CDCPD tracking moves all the points of the rope to the right side, effectively shrinking the object to fit it to the visible points.

To address this problem, we introduce a novel geometric prediction-based regularization term into the GMM-EM process. By using a geometric prediction like [24], [25], [26], we are able to avoid the downsides of using a physics simulation while capturing much of the behavior of a deformable object. Sec. V-A briefly details the models that we use in our results, but our method is agnostic to the specifics of the model used.

Let $\mathcal{P}^{\text{pred}}$ be a prediction of the configuration of the object made by some model of deformable object motion. We define an additional cost function

$$\begin{aligned} E_{PRED}(W) &= \sum_{m=1}^{M} \left\| p_m^{\text{GMM}} - p_m^{\text{pred}} \right\|^2 \\ &= \text{Tr}\big((\mathcal{P}^{\text{GMM}} - \mathcal{P}^{\text{pred}})^T (\mathcal{P}^{\text{GMM}} - \mathcal{P}^{\text{pred}})\big) \\ &= \text{Tr}\big((\mathcal{P}^{\text{prev}} + GW)^T (\mathcal{P}^{\text{prev}} + GW) \\ &\quad - 2(\mathcal{P}^{\text{prev}} + GW)^T \mathcal{P}^{\text{pred}} + \mathcal{P}^{\text{pred}^T} \mathcal{P}^{\text{pred}}\big) \end{aligned} \tag{12}$$

to encode the information from the model. Below we show how to incorporate Eq. 12 in the CDCPD regularization procedure.

Thus, similar to a tracking method like the Kalman filter, we make a prediction with the motion model, and then use it to update our tracking result along with what we observe at the next frame. This approach has the advantage of not being too sensitive to large perceptual changes (e.g. the end of the rope disappearing), because the motion model helps to retain physical plausibility.

4) Solving for the Transform Weights: Adding all three regularization terms, we get the final cost function for the M-step:

$$\begin{aligned} &Q(W, \sigma^2) = \\ &\sum_{n=1}^{N} \sum_{m=1}^{M} \mathsf{P}(m|d_n) \frac{\left\| d_n - (p_m^{\text{prev}} + G(m,\cdot)W \right\|^2}{2\sigma^2} \\ &+ \frac{3N_{\mathsf{P}}}{2} \log(\sigma^2) + \frac{\alpha}{2} E_{CPD}(W) \\ &+ \frac{\gamma}{2} E_{LLE}(W) + \frac{\zeta}{2} E_{PRED}(W), \end{aligned} \tag{13}$$

where α, γ, and ζ are constant weights and $N_{\mathsf{P}} = \sum_{n,m=1}^{N,M} \mathsf{P}(m|d_n)$.

Similar to the process shown in [23], we can minimize Q in the M-step by computing a W where $\partial Q / \partial W = 0$: Denote $\mathbf{1}$ as a column vector of ones, and $d(v)$ as the diagonal form of vector v. Let $H = (\mathbf{I} - L)^T (\mathbf{I} - L)$. Then,

$$\begin{aligned} A &= \big(d(X\mathbf{1})G + \alpha\sigma^2 I + \gamma\sigma^2 HG + \zeta G\big) \\ B &= X\mathcal{D} - (d(X\mathbf{1}) + \gamma\sigma^2 H)\mathcal{P}^{\text{GMM}} + \zeta(\mathcal{P}^{\text{pred}} - \mathcal{P}^{\text{GMM}}), \end{aligned} \tag{14}$$

where A is a $M \times M$ matrix and B is a $M \times 3$ matrix. Both A and B are calculated directly from terms we have already computed. We then obtain W by solving

$$AW = B. \tag{15}$$

Then we can obtain σ^2 using Eq. (19) from [7]:

$$\begin{aligned} \sigma^2 = &\frac{1}{3N_{\mathsf{P}}} \left(\text{Tr}\left(\mathcal{D}^T d(X^T\mathbf{1})\mathcal{D}\right) \right) - 2\,\text{Tr}\left((\mathcal{P}^{\text{GMM}})^T X\mathcal{D}\right) \\ &- 2\,\text{Tr}\left(W^T G^T X\mathcal{D}\right) + \text{Tr}\left((\mathcal{P}^{\text{GMM}})^T d(X\mathbf{1})\mathcal{P}^{\text{GMM}}\right) \\ &+ 2\,\text{Tr}\left(W^T G^T d(X\mathbf{1})\mathcal{P}^{\text{GMM}}\right) + \text{Tr}\left(W^T G^T d(X\mathbf{1})GW\right) \end{aligned} \tag{16}$$

We repeat the E-step and M-step until it reaches the maximum number of iterations (100) or the change in σ^2 between two iterations is smaller than a threshold (10^{-4}). $\mathcal{P}^{\text{GMM}}$ after one time step is thus $\mathcal{P}^{\text{prev}} + GW$.

C. Posterior Constraints

GMM-EM is able to reason about probabilities and costs, but the result is not guaranteed to be *geometrically consistent*. For example, points could be further apart than is possible for a given deformable object. To address this, CDCPD introduced a post-processing optimization step which enforces some constraints. We start with those constraints (Sec. IV-C.1) and propose novel convex constraints for preventing self-intersection (Sec. IV-C.2) and obstacle penetration (Sec. IV-C.3). We emphasize that using convex constraints is essential for efficient and reliable optimization.

Denote p_m^{GMM} as the mth point of the GMM-EM result $\mathcal{P}^{\text{GMM}} = \mathcal{P}^{\text{prev}} + GW$. Then our goal is to adjust every p_m^{GMM} such that all points comply with the geometric constraints described above.

1) Known Correspondences and Stretching Limits: CDCPD considers two constraints. First, a constraint based on the maximum distance between the nodes in the mesh

$$\|p_i - p_j\| \leq \lambda \rho_{ij} \quad \forall (i,j) \in \mathcal{E} \tag{17}$$

with a parameter $\lambda \geq 1$ which controls the flexibility of the constraint. Second, a constraint based on *known correspondences*. For example, if we know that the grippers are rigidly connected to a particular part of the deformable object, then we can encode that constraint directly into the optimization at each timestep. We represent these correspondences between a set of known points $\{z_1, \ldots, z_K\} \subset \mathbb{R}^{3K}$ and corresponding deformable point indices as $\mathcal{C}_{\text{idx}} = [c_1, c_2, \ldots, c_K] \in \mathbb{N}^{2K}$

$$p_m = z_k \quad \forall (m,k) \in \mathcal{C}_{\text{idx}} \ . \tag{18}$$

2) Self-Intersection Constraints: The first novel constraint we add is designed to prevent a tracking result from passing through itself (see Fig. 3). The general idea of preventing self intersection is checking pairs of edges for potential intersection and then constraining the movement of the points such that a small gap remains between any edges that could cross. This gap corresponds to the thickness of the object. Specifically, our algorithm checks all pairs of edges that do not share points. For two edges $e_i = (i_0, i_1), e_j = (j_0, j_1) \in \mathcal{E}$, we regard them as two line segments and calculate the shortest distance s_{ij} between the two closest points, $p_{\text{near},i}$ and $p_{\text{near},j}$, on these line segments. These points are:

$$p^{\text{prev}}_{\text{near},i} = r_i \mathcal{P}^{\text{prev}}(i_0) + (1 - r_i)\mathcal{P}^{\text{prev}}(i_1) \tag{19}$$

$$p^{\text{prev}}_{\text{near},j} = r_j \mathcal{P}^{\text{prev}}(j_0) + (1 - r_j)\mathcal{P}^{\text{prev}}(j_1) \ , \tag{20}$$

where r_i and r_j are two real number between 0 and 1. If s_{ij} is smaller than the threshold s_{check} we will constrain $\mathcal{P}(i_0)$, $\mathcal{P}(i_1)$, $\mathcal{P}(j_0)$ and $\mathcal{P}(j_1)$. The larger s_{check} is, the less likely it is that we miss a self-intersection between two frames. However, a large s_{check} will likely result in more constraints in the optimization problem, which leads to a higher computation cost and potentially an infeasible optimization problem.

The closest points $p^{\text{prev}}_{\text{near},i}$ and $p^{\text{prev}}_{\text{near},j}$ found in $\mathcal{P}^{\text{prev}}$ above correspond to the closest points $p_{\text{near},i}$ and $p_{\text{near},j}$ found in $\mathcal{P}$. $p_{\text{near},i}$ and $p_{\text{near},j}$ are calculated as below:

$$p_{\text{near},i} = r_i \mathcal{P}(i_0) + (1 - r_i)\mathcal{P}(i_1) \tag{21}$$

$$p_{\text{near},j} = r_j \mathcal{P}(j_0) + (1 - r_j)\mathcal{P}(j_1) \tag{22}$$

We constrain $\mathcal{P}$ by requiring the projection of vector $p_{\text{near},i} - p_{\text{near},j}$ on the direction of $p^{\text{prev}}_{\text{near},i} - p^{\text{prev}}_{\text{near},j}$ to be larger than a collision threshold s. We write the constraints as below:

$$(p_{\text{near},i} - p_{\text{near},j})^T \frac{p^{\text{prev}}_{\text{near},i} - p^{\text{prev}}_{\text{near},j}}{\left\| p^{\text{prev}}_{\text{near},i} - p^{\text{prev}}_{\text{near},j} \right\|} > s \quad \forall (i,j) \in \mathcal{E} \tag{23}$$

Intuitively, the parameter s determines how far edges should keep away from each other, i.e. it approximates object thickness.

3) Obstacle Interaction Constraints: In addition to self-intersection constraints, we include a novel constraint to account for obstacle interaction. If we know the geometry of some obstacles in the scene, we can constrain the result of the tracking process to stay consistent with that known geometry. To achieve this, we add constraints based on the local geometry near each point p^{prev}_m. For each point p^{prev}_m, we find the nearest obstacle point o_m and the corresponding normal vector n_m. We then constrain the tracking result to stay on the same side of the tangent plane defined by o_m and n_m: $(p_m - o_m)^T n_m > 0 \quad \forall p_m \in \mathcal{P}$.

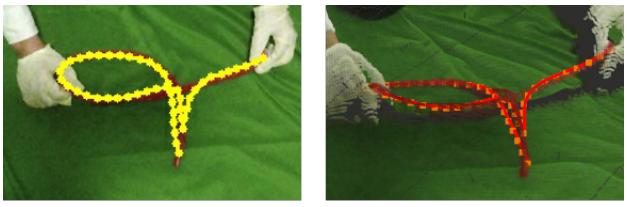

Fig. 3: Example of CDCPD tracking result passing through itself when raised.

Combining all the constraints together yields the following convex optimization problem:

$$\begin{aligned} \underset{\mathcal{P}}{\text{argmin}} \quad & \sum_{m=1}^{M} \left\| p_m - p_m^{\text{GMM}} \right\|^2 \\ \text{subject to} \quad & \|p_i - p_j\| \leq \lambda \rho_{ij} \quad \forall (i,j) \in \mathcal{E} \\ & p_m = z_k \quad \forall (m,k) \in \mathcal{C}_{\text{idx}} \\ & (p_{\text{near},i} - p_{\text{near},j})^T \frac{p^{\text{prev}}_{\text{near},i} - p^{\text{prev}}_{\text{near},j}}{\left\| p^{\text{prev}}_{\text{near},i} - p^{\text{prev}}_{\text{near},j} \right\|} > s \\ & (p_m - o_m)^T n_m > 0 \quad \forall m \in [1, \ldots, M] \ , \end{aligned} \tag{24}$$

which we solve with the Gurobi [27] optimization package, yielding our final estimate of $\mathcal{P}$.

V. Experiments

We conducted experiments in simulation and the real world to analyze our algorithm quantitatively and qualitatively in the presence of severe occlusion and obstacles (see the accompanying video). We compare our results with CPD+physics [10] and CDCPD [7].

The parameter values are: $\beta = 1.0$, $\alpha = 0.5$, $\gamma = 1.0$, $\zeta = 2.0$, $k_{vis} = 100$ (see Eq. (7) of [7]), $s_{check} = 0.02m$, $s = 0.01m$, and $\lambda = 1.1$. Point clouds are downsampled using a voxel grid filter with a grid size of 2cm. CDCPD2, CDCPD and CPD+Physics are implemented in C++ and tested on an Intel i7-8700 @ 3.7GHz processor with 32 GB RAM.

A. Geometric Prediction Models

Because our method relies on a prediction of deformable object motion, we need an efficient model that outputs this kind of prediction. We emphasize that the model need not be very accurate, but only that it is more physically plausible than the output of the CPD process alone. To show our algorithm's performance with (and robustness to) a variety of models, we provide results for three different geometric models of motion. First, the most naive model, we call *No Motion*, assumes there is no movement between frames: $\mathcal{P}^{\text{pred}} = \mathcal{P}^{\text{prev}}$. While naive, this model may be reasonable when the movement between two frames is small. The second prediction model we evaluate is the *diminishing rigidity model* [24], [25]: $\mathcal{P}^{\text{pred}} = \mathcal{P}^{\text{prev}} + J(\mathcal{P}, q)\dot{q}$, where $J(\mathcal{P}, q)$ is an estimate of the Jacobian mapping gripper movements to deformable object movement. We use $k = 10.0$ (see [24]). The third model prediction we tried is

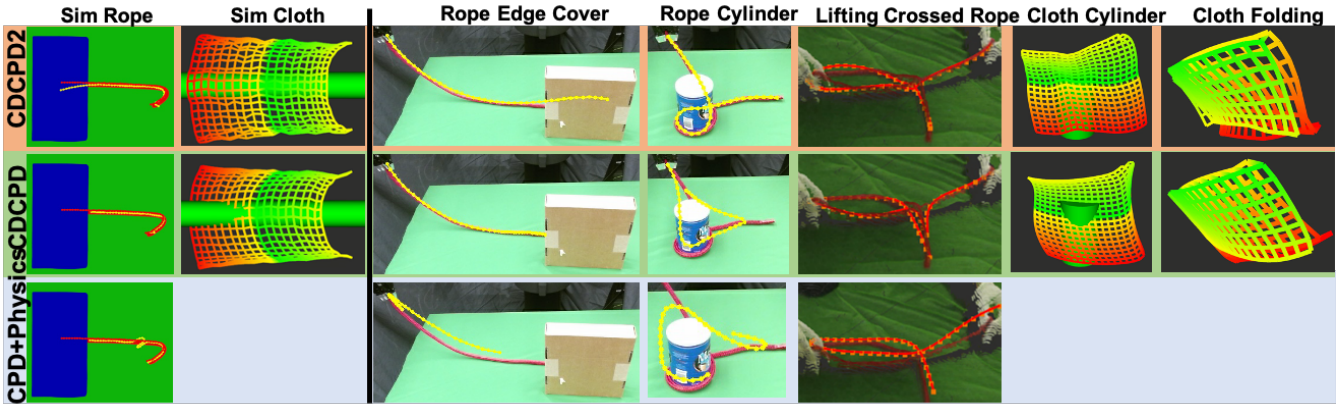

Fig. 4: Comparison of results of CDCPD2 and baselines for simulation (left of line) and real (right of line) data. The CPD+Physics implementation from the authors of [10] does not track cloth, so it is used only for rope. Columns show results for a single frame in each task. First column: ground truth (red) vs. estimates (yellow). CDCPD2 used the diminishing rigidity model for all tasks shown except lifting crossed rope, where we used the "no motion" model because the rope was manipulated by a human (so $\dot{q}$ is unknown).

a *constrained directional rigidity model* [26]. It builds on diminishing rigidity by directly accounting for the direction of gripper motion, and formulating constraints to account for obstacles. This results in a more expressive Jacobian: $\mathcal{P}^{\text{pred}} = \mathcal{P}^{\text{prev}} + J(\mathcal{P}, q, \dot{q}, \mathcal{O})\dot{q}$. We use $k_r = 10.0$, $k_g = 5.0$ and $k_D = 5.0$ (see [26]).

B. Experiments with Simulated Data

To analyze performance quantitatively, we obtained ground truth from simulation in Blender and compared all methods to this. The rope is modelled as a soft body with 49 line segments. The cloth is modelled as a 20 by 20 mesh. RGBD images are rendered using the Eevee rendering engine with size 810 $\times$ 540.

The first experiment demonstrates the ability of our method to prevent the tracking result from shrinking to the visible part of the object (as in Fig. 2). We drag one end of the rope until the free end of the rope is occluded by the obstacle. We can see from Fig. 4 that our result won't shrink (regardless of the motion model used), while CDCPD and CPD+Physics will shrink quickly. Fig. 5 shows the comparison with ground truth. We can see when the free end is not occluded, i.e. the first several frames, the error of all these methods is close. However, when the free end is occluded, the error of CDCPD and CPD+Physics becomes much larger.

Our second experiment demonstrates our algorithm's ability to handle interaction with obstacles, Here we drape a cloth over a cylinder with a camera looking from above. Our result doesn't penetrate the cylinder, while CDCPD's does, as shown in Fig. 4.

C. Experiments with real data

We performed several experiments with real data to guage the qualitative performance of our method. Examples are shown in Fig. 4 and the accompanying video. The experiments showed manipulation of a rope and cloth under occlusion and interacting with obstacles. Due to space limitations we do not describe each experiment in detail but instead present the key observations: 1) For both rope and cloth, our method was robust to edge-covering (i.e. not shrinking when an edge was covered), unlike CDCPD and CPD+Physics; 2) CDCPD2 was effective in keeping the object out of known obstacles (e.g. Fig. 1) while CDCPD was not; 3) CDCPD2 prevented edge crossings for rope on-par with CPD+Physics and better than CDCPD; 4) For many experiments, the choice of motion model did not have a major impact, however the best-performing model overall was diminishing rigidity.

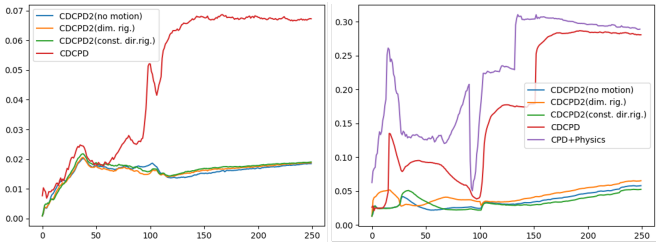

Fig. 5: Mean distance error (m) vs. frame index for simulation experiments. Left: cloth covering cylinder. Right: rope edge-covering.

D. Computation time

The average computation time per frame of CDCPD2 across all tasks is 26 ms (rope) and 193 ms (cloth). For CDCPD, the average computation time is 22 ms (rope) and 101 ms (cloth). For CPD+Physics, the average computation time is 23 ms (rope). The increase in computation time for our method on cloth is largely due to the additional obstacle interaction and self-intersection constraints, which take time to compute and entail a more difficult optimization problem.

VI. Conclusion

Our results show the ability of our algorithm to handle obstacle interaction, self-intersection and severe occlusion when tracking deformable objects better than previous work. We increased the robustness to occlusion by introducing a prediction-based regularization term in GMM-EM, which is inspired by the Kalman filter. By adding posterior constraints to the result from GMM-EM, we prevented the result from penetrating obstacles and intersecting itself. In future work, we may explore learning a good initialization of W by training on a dataset collected in a simulation environment.

References

[1] R. White, K. Crane, and D. A. Forsyth, "Capturing and animating occluded cloth," *ACM Trans. Graph.*, vol. 26, no. 3, July 2007.

[2] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 175:1–175:10, Dec. 2009.

[3] T. Collins, A. Bartoli, N. Bourdel, and M. Canis, "Robust, real-time, dense and deformable 3d organ tracking in laparoscopic videos," in *Medical Image Computing and Computer-Assisted Intervention*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds. Cham: Springer International Publishing, 2016, pp. 404–412.

[4] N. Haouchine, J. Dequidt, I. Peterlik, E. Kerrien, M. Berger, and S. Cotin, "Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery," in *International Symposium on Mixed and Augmented Reality*, Oct 2013, pp. 199–208.

[5] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *CVPR*, June 2015, pp. 343–352.

[6] D. Pizarro and A. Bartoli, "Feature-based deformable surface detection with self-occlusion reasoning," *International Journal of Computer Vision*, vol. 97, no. 1, pp. 54–70, 2012.

[7] C. Chi and D. Berenson, "Occlusion-robust deformable object tracking without physics simulation," in *IROS*, 2019, pp. 6443–6450.

[8] T. Tang and M. Tomizuka, "Track deformable objects from point clouds with structure preserved registration," *The International Journal of Robotics Research*, p. 0278364919841431, 2018.

[9] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3426–3433, 2018.

[10] T. Tang, Y. Fan, H. Lin, and M. Tomizuka, "State estimation for deformable objects by point registration and dynamic simulation," in *IROS*, Sep. 2017, pp. 2427–2433.

[11] N. Padoy and G. Hager, "Deformable tracking of textured curvilinear objects," *BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012*, 01 2012.

[12] A. Petit, V. Lippiello, and B. Siciliano, "Real-time tracking of 3d elastic objects with an rgb-d sensor," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3914–3921.

[13] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2308–2315.

[14] T. Matsuno and T. Fukuda, "Manipulation of flexible rope using topological model based on sensor information," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2638–2643.

[15] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 371–395, 2006.

[16] W. H. Lui and A. Saxena, "Tangled: Learning to untangle ropes with rgb-d perception," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 837–844.

[17] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1130–1137.

[18] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[19] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5431–5440.

[20] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[21] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[22] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec 2010.

[23] S. Ge, G. Fan, and M. Ding, "Non-rigid point set registration with global-local topology preservation," in *CVPR Workshops*, June 2014, pp. 245–251.

[24] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *IROS*, Nov 2013, pp. 4525–4532.

[25] D. Mcconachie and D. Berenson, "Estimating model utility for deformable object manipulation using multiarmed bandit methods," *T-ASE*, vol. 15, no. 3, pp. 967–979, July 2018.

[26] M. Ruan, D. M. Conachie, and D. Berenson, "Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation," in *IROS*, Oct 2018, pp. 512–519.

[27] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com