Tracking Deformable Objects with Point Clouds

John Schulman, Alex Lee, Jonathan Ho, Pieter Abbeel

Abstract— **We introduce an algorithm for tracking deformable objects from a sequence of point clouds. The proposed tracking algorithm is based on a probabilistic generative model that incorporates observations of the point cloud and the physical properties of the tracked object and its environment. We propose a modified expectation maximization algorithm to perform maximum a posteriori estimation to update the state estimate at each time step. Our modification makes it practical to perform the inference through calls to a physics simulation engine. This is significant because (i) it allows for the use of highly optimized physics simulation engines for the core computations of our tracking algorithm, and (ii) it makes it possible to naturally, and efficiently, account for physical constraints imposed by collisions, grasping actions, and material properties in the observation updates.**

Even in the presence of the relatively large occlusions that occur during manipulation tasks, our algorithm is able to robustly track a variety of types of deformable objects, including ones that are one-dimensional, such as ropes; two-dimensional, such as cloth; and three-dimensional, such as sponges. Our implementation can track these objects in real time.

I. Introduction

Most objects we interact with in our everyday lives, such as food, clothing, paper, and living creatures, are deformable. In order to teach robots to make us breakfast, tidy up our rooms, or assist in surgical operations, we need to enable them to perceive and control these objects in unstructured environments.

Due to the high dimensionality of the state spaces of deformable objects, perceiving deformable objects is much more difficult than perceiving rigid objects. Often, self-occlusions make it impossible to infer the full states of deformable objects from a single view. In addition, many deformable objects of interest lack distinguishable key-points.

We present a new probabilistic approach for tracking deformable objects, inspired by (i) New high frame-rate 3D sensory capabilities provided by RGB-D cameras such as the Microsoft Kinect and by (ii) Advances in algorithms for computer simulation of physics and in modern computational capabilities, which together enable real-time simulation of deformable objects in general-purpose physics simulators. Our approach defines a probabilistic model for the object and environment dynamics, as well as for the sensory measurements, and we provide an efficient algorithm to perform the probabilistic inference in this model.

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94709.

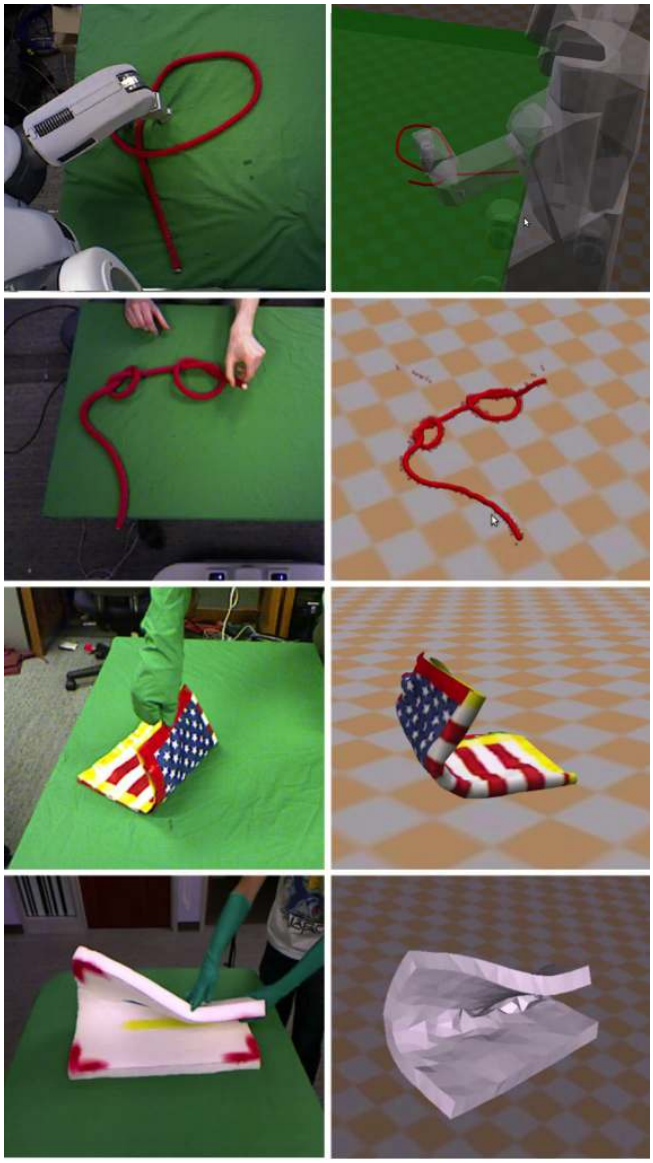

Fig. 1: The left panels show camera images of the tracked objects, and the right panels show renderings of our state estimates. Note that the viewpoints are different; our algorithm infers the 3D configuration of the object.

A summary of our contributions is as follows:

- A new probabilistic observation model that addresses unknown point correspondences and occlusions.
- A modified EM algorithm to perform inference in this model: We show how the probabilistic inference for finding the maximum a posteriori estimate of the state at each time step can be performed through calls to the physics-based simulation. This is significant because (i)

it allows for the use of highly optimized physics simulation engines for the core computations of our tracking algorithm, and (ii) it makes it possible to naturally, and efficiently, account for physical constraints imposed by collisions, grasping actions, and material properties in the observation updates.
- An implementation that can successfully track a variety of extremely deformable objects in real time as they are manipulated by a human or a robot. To the authors' knowledge, we present the first results of tracking extremely deformable objects such as rope and cloth under arbitrary manipulations (including knot tying and folding). Our implementation runs in real time and is suitable for closed-loop control in robotic manipulation tasks. Videos of the results can be found at `http://rll.berkeley.edu/tracking`, along with source code and our ground truth dataset (described in Section IX.)

II. Previous work

The idea of using physical models to estimate the state of objects in images dates back to the influential papers by Kass, Terzopoulous, and Witkin [10, 19]. They defined energy functions for parametric curves and surfaces; these energy functions contained an "external" energy that attracted the curves and surfaces to image features and an "internal" energy that penalized excessive curvature or deformation, providing regularization. These methods, called active contours and active surfaces (or, equivalently, snakes) have been widely used in computer vision, for example, to find the outlines of objects in images and video. Metaxas and Terzopoulos proposed an algorithm for tracking deformable objects that had a particular parameterization (linked superquadrics) [11]. Their approach provides a second-order dynamics model for these objects and uses a Extended Kalman Filter to update the positions and covariances of the objects' degrees of freedom.

Tracking is closely related to registration, where the typical problem is to find a nonlinear warping function that maps one surface, image, or point cloud onto another. The last two decades have seen a lot of progress in registration of points and volumes, motivated by problems in 3D modeling and medical image analysis.

The simplest algorithm for registration of point clouds, assumed to come from a rigid object, is the iterative closest point (ICP) algorithm [3, 1]. The ICP algorithm alternates between (1) calculating correspondences between points and (2) solving for a transformation that maximally aligns the corresponding points. The ICP algorithm has been generalized to allow for transformations other than rigid motions. Chui and Rangarajan [4] alternated fitting a non-rigid transformation with a soft-assignment correspondence calculation. Hahnel, Thrun, and Burgard seek to reconstruct the geometry of non-stationary, non-rigid objects; they endow the acquired point clouds with a link structure and register pairs of scans using an ICP-like optimization [8].

A variety of other work addresses problems in registration, state estimation, and tracking of deformable objects. One important and well-studied problem is the non-rigid registration of 3D medical images. Murphy et al. [13] give an assessment of modern approaches to this problem. Saltzmann et al. [17] provide an algorithm to track a deforming surface; their approach uses a reduced-dimensionality parametrization of a textured surface and relies on keypoint matches. De Aguiar et al. [5] track the human body based on video data from multiple cameras using a 3D textured mesh model and optimizing a set of control points in each video frame using an optical flow-based method. Miller et al. [12] visually infer the state of a folded piece of cloth by optimizing a geometric model of folded clothing that mostly relies on the contour.

III. Probabilistic model for observations

This section describes our probabilistic model for the process by which a tracked deformable object generates observations in the form of a point cloud. Each tracked object has a corresponding physical model.[1] As is common in computer-based simulation of such objects, we assume this physical model has a set of K points with 3D coordinates $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$, which we will abbreviate as $\mathbf{m}_{1:K}$, that completely define the current configuration of the object. These control points could be on the backbone of a rope, or they could be a set of vertices of a triangulation of a piece of cloth, or a set of vertices of a tetrahedral mesh of a sponge, respectively (see Figure 2 for illustration). At each time step, the sensor hardware generates from the visible portion of the object a point cloud consisting of N 3D points. We use $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$, abbreviated by $\mathbf{c}_{1:N}$, to refer to the coordinates of the points in the point cloud.

In Section V, we consider the generalization where each of the K nodes and each of the N points in the point cloud is associated with a $d-$dimensional feature vector ($d \geq 3$) that locally characterizes the object. The feature vector's components include position and optionally color and texture descriptors. However, in the following discussion, $\mathbf{m}_i \in \mathbb{R}^3$ and $\mathbf{c}_i \in \mathbb{R}^3$.

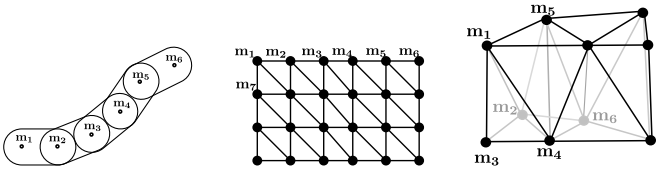

Fig. 2: Physical model for three classes of deformable object: (a) one-dimensional object, which we model as a chain, (b) two-dimensional object, which we model as a mass-spring system on a triangular mesh, (c) three-dimensional object, which we model as a mass-spring system on a tetrahedral mesh.

[1] Our approach does not depend on the specifics of the physical model used for simulation. The physical models we used in our experiments were chosen based on accuracy under real-time constraints and availability in state-of-the-art physics-based simulation engines. We used Bullet, which uses fairly standard models and methods for real-time simulation. Rope is simulated as a chain of linked capsules with bending and torsional energy at the joints. Cloth and sponge are simulated as mass spring systems on triangular and tetrahedral meshes.

At each time step, our goal is to infer the values of $\mathbf{m}_{1:K}$ given the point cloud $\mathbf{c}_{1:N}$. The correspondence between the point cloud and the model's control points is not known and must be inferred. These correspondences are encoded in the "responsibility" variables z_{kn} which indicate if the object surface nearest to $\mathbf{m}_k$ generated observation $\mathbf{c}_n$:

$$z_{kn} = \mathbb{1}_{[\mathbf{m}_k \text{ is responsible for } \mathbf{c}_n]}. \tag{1}$$

We also include a source of "noise points" in our probabilistic model, which are not generated by the tracked object. Noise points may occur because of errors in background subtraction and incorrect registration between color and depth images. The variable $z_{K+1,n}$ indicates if the nth point is noise:

$$z_{K+1,n} = \mathbb{1}_{[\mathbf{c}_n \text{ is a noise point}]}. \tag{2}$$

The probability distribution of noise points is uniform throughout space with density ρ_{noise}

$$p(\mathbf{c}_n | z_{K+1,n} = 1) = \rho_{noise} \tag{3}$$

for some parameter ρ_{noise}. (Note that this distribution is not normalized.) The following heuristic choice of ρ_{noise} is empirically good: $\rho_{noise} = .05\,\mathrm{cm}^{-3}$.

The visibility or occlusion of the object is described by the visibility variables, v_k, $k = 1, 2, \ldots, K$, where

$$v_k = \mathbb{1}_{[\mathbf{m}_k \text{ is visible from the camera}]}. \tag{4}$$

Explicitly, our probabilistic generative model assumes the data generation process is as follows:

- For each $n \in 1, 2, \ldots, N$, an index $k_n \in 1, 2, \ldots, K+1$ is selected to be responsible for that point, where $p(k_n = k) \propto v_k$, i.e.,

$$p(k_n = k) = \begin{cases} \frac{v_k}{\sum_{k'} v_{k'}+1} & \text{for } k = 1, 2, \ldots, K \\ \frac{1}{\sum_{k'} v_{k'}+1} & \text{for } k = K+1 \end{cases} \tag{5}$$

  The responsibility variables z_{kn} represent this selection as follows

$$z_{kn} = \begin{cases} 1 & \text{if } k = k_n \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- $\mathbf{c}_n$ is normally distributed $\mathbf{c}_n \sim \mathcal{N}(\mathbf{m}_k, \Sigma_k)$ if point n is not noise, else $\mathbf{c}_n$ is distributed with uniform density ρ_{noise} throughout space.
- Each node is associated with a diagonal covariance matrix $\Sigma_k = \mathrm{diag}((\sigma^x)^2, (\sigma^y)^2, (\sigma^z)^2)$, where the σ are chosen to be approximately the distance between nodes on the object.

Lastly, we put a prior on $\mathbf{m}_{1:K}$:

$$p(\mathbf{m}_{1:K}) = e^{-\frac{1}{\eta} V_0(\mathbf{m}_{1:K})} \tag{7}$$

where $V_0(\mathbf{m}_{1:K})$ is the potential energy of the tracked object, which includes gravitational potential energy and bending energy. $V_0(\mathbf{m}_{1:K}) = \infty$ for disallowed states, e.g., states where solid objects overlap. Using this prior V_0 corresponds to assuming a quasi-static scenario where objects move slowly and are continually at low-energy configurations.

The probabilistic graphical model is depicted in Figure 3. Without the visibility variables and the prior on $\mathbf{m}$, this model is a mixture of Gaussians model. However, a fundamentally different inference procedure is necessary due to the need for incorporation of the physical properties and constraints in this problem.

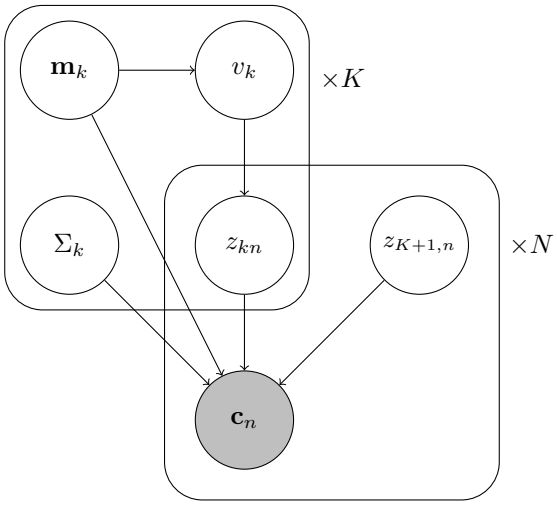

Fig. 3: Graphical model for the observed point cloud

The probability of the state at time $t+1$, $\mathbf{m}_{1:K}^{t+1}$, has no explicit dependence on the previous estimate $\mathbf{m}_{1:K}^{t}$. However, the inference algorithm at time $t+1$ will be initialized using $\mathbf{m}_{1:K}^{t}$, and the optimization (described below) will converge to the local maximum of probability $\log p(\mathbf{m}_{1:K}^{t+1} | \mathbf{c}_{1:N}^{t+1})$ whose basin of attraction includes $\mathbf{m}_{1:K}^{t}$. This corresponds to assuming quasi-static dynamics. This scheme will ensure, for example, that a tied knot will remain tied, that rigid objects can't intersect, etc., as well as prefer low-energy states.

IV. Inference

For each observation (point cloud), our algorithm's task is to find the most probable node positions given the measurement, i.e., to calculate

$$\arg\max_{\mathbf{m}_{1:K}} p(\mathbf{m}_{1:K} | \mathbf{c}_{1:N}). \tag{8}$$

To solve this maximum a posteriori (MAP) estimation problem, we use the expectation maximization (EM) algorithm. The EM algorithm is typically used in the setting of maximum likelihood estimation, but it can just as well be used for MAP estimation. In the present problem, the distinction is important because we put a prior on the state $p(\mathbf{m}_{1:K})$ that penalizes, e.g., floating objects and excessive bending. The EM algorithm alternates between generating a lower bound to the log-probability function based on an expectation over the latent variables (E step) and maximizing this lower bound with respect to the node positions (M step) [14, 6].

For self-containedness, and to be able to precisely articulate our contributions that made the probabilistic inference

practical in our setting, we will review the EM algorithm formulation in the setting of a general probabilistic model (with notation suggestive to the current problem), and then we'll describe its specialization to the current problem in the following two subsections.

Let $\mathbf{M}$ denote the random variables we are trying to infer, let $\mathbf{C}$ denote the observed variables, and let $\mathbf{Z}$ denote some other latent variables that we will marginalize over. Our aim is to find the most probable $\mathbf{M}$:

$$\arg\max_{\mathbf{M}} \log p(\mathbf{M}, \mathbf{C}) = \log p(\mathbf{C}|\mathbf{M}) + \log p(\mathbf{M}), \quad (9)$$

which is equivalent to maximizing $p(\mathbf{M}|\mathbf{C})$. We exploit the following identity, which holds for an arbitrary distribution $q(z)$:

$$\log p(\mathbf{M}, \mathbf{C}) = \mathcal{L}_q(\mathbf{M}) + KL(q|p) + \log p(\mathbf{M}) \quad (10)$$

where

$$\mathcal{L}_q(\mathbf{M}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left(\frac{p(\mathbf{C}, \mathbf{Z}|\mathbf{M})}{q(\mathbf{Z})} \right) \quad (11)$$

$$KL(q|p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left(\frac{p(\mathbf{Z}|\mathbf{C}, \mathbf{M})}{q(\mathbf{Z})} \right). \quad (12)$$

$\mathcal{L}_q(\mathbf{Z})$ serves as our lower bound to the log-likelihood function $\log p(\mathbf{C}|\mathbf{M})$ since the Kullback-Leibler divergence $KL(q|p)$ is always positive. The ith iteration is as follows:

- E step: $q^{(i)}(\mathbf{Z}) \leftarrow p(\mathbf{Z}|\mathbf{C}, \mathbf{M}^{(i-1)})$.
- M step: $\mathbf{M}^{(i)} \leftarrow \arg\max_{\mathbf{M}} \left[\mathcal{L}_{q^{(i)}}(\mathbf{M}) + \log p(\mathbf{M})\right]$

The actual objective $\log p(\mathbf{M}, \mathbf{C})$ is guaranteed not to decrease

$$\log p(\mathbf{M}^i, \mathbf{C}) \leq \log p(\mathbf{M}^{i+1}, \mathbf{C}) \quad i = 1, 2, 3, \ldots \quad (13)$$

and it is unchanged only if both $\mathcal{L}_q$ and $\mathbf{M}$ are unchanged in an iteration.

E step

In the observation model considered in this paper, the E step computes $p(z_{kn}|\mathbf{c}_{1:N}; \mathbf{m}_{1:K})$. First we compute the visibility variables v_k, which are a deterministic function of the scene and the depth image. Specifically, a ray-casting operation detects if the line segment from the camera to $\mathbf{m}_k$ is blocked by some modeled object (e.g., the robot or the tracked object itself). We also check if the depth image contains a point significantly in front of $\mathbf{m}_i$.

$$1 - v_k = \mathbb{1}\big[\mathbf{m}_i \text{ is occluded by a modeled object} \cup \text{depth}(i, j) < \|\mathbf{m}_i\| - 3\,\text{cm}\big]. \quad (14)$$

Then we apply Bayes rule to calculate the posterior probability of the correspondence variables:

$$p(z_{kn} = 1|\mathbf{m}_{1:K}, \mathbf{c}_{1:N}) \quad (15)$$

$$= \frac{p(\mathbf{c}_n|z_{kn} = 1, \mathbf{m}_k)p(z_{kn} = 1)}{p(\mathbf{c}_n)} \quad (16)$$

$$= \frac{p(\mathbf{c}_n|z_{kn} = 1, \mathbf{m}_k)v_k}{\sum_{k'=1}^{K} p(\mathbf{c}_n|z_{k'n} = 1, \mathbf{m}_k)v_{k'} + \rho_{noise}} \quad (17)$$

$$= \frac{\mathcal{N}(\mathbf{c}_n; \mathbf{m}_k, \Sigma_k)v_k}{\sum_{k'=1}^{K} \mathcal{N}(\mathbf{c}_n; \mathbf{m}_{k'})v_{k'} + \rho_{noise}}. \quad (18)$$

The log-probability lower bound is calculated as follows:

$$\mathcal{L}_q(\mathbf{m}_{1:K}) = \sum_{\mathbf{Z}} p(\mathbf{Z}) \log p(\mathbf{c}_{1:N}|\mathbf{m}_{1:K}, \mathbf{Z}) \quad (19)$$

$$= \sum_{\mathbf{Z}} p(\mathbf{Z}) \log \prod_n \prod_k p(\mathbf{c}_n|\mathbf{m}_k, z_{kn} = 1)^{z_{kn}} \quad (20)$$

$$= \sum_{\mathbf{Z}} p(\mathbf{Z}) \sum_n \sum_k z_{kn} \log p(\mathbf{c}_n|\mathbf{m}_k, z_{kn} = 1) \quad (21)$$

$$= \sum_n \sum_k \alpha_{nk} \log \mathcal{N}(\mathbf{c}_n; \mathbf{m}_k, \Sigma_k) \quad (22)$$

where

$$\alpha_{nk} = \mathbb{E}[z_{kn}|\mathbf{m}_{1:K}, \mathbf{c}_{1:N}] = \log p(z_{kn} = 1|\mathbf{m}_{1:K}, \mathbf{c}_{1:N}) \quad (23)$$

Finally, the total log-probability lower bound (Equation 10) is

$$\log p(\mathbf{m}_{1:K}, \mathbf{c}_{1:N}) = \log p(\mathbf{m}_{1:K}) + \log p(\mathbf{c}_{1:N}|\mathbf{m}_{1:K}) \quad (24)$$

$$= V_0(\mathbf{m}_{1:K}) + \sum_n \sum_k \alpha_{nk} \log \mathcal{N}(\mathbf{c}_n; \mathbf{m}_k, \Sigma_k). \quad (25)$$

M step

The M step solves the optimization problem

$$\arg\max_{\mathbf{m}_{1:K}} \left[\mathcal{L}_q(\mathbf{m}_{1:K}) + \log p(\mathbf{m}_{1:K})\right] \quad (26)$$

This is a hard optimization problem due to the numerous non-convex costs and constraints in our physics-based prior, such as non-penetration of solids and resistance to stretching and bending. In this section we describe how this hard optimization problem can be solved efficiently by taking advantage of physics-based simulation engines.

The second term in Equation (26) corresponds exactly to the (negative of the) potential energy of our system. Simply running the physics simulation would lead the system to a local minimum in potential energy (since the total energy monotonically decreases due to damping) while respecting all of the physical constraints, but would fail to account for the first term in Equation (26), which is there to account for the observations.

To make the physics-based simulation engine account for the first term, we impose an artificial "observation" potential energy $V^{obs}(\mathbf{m}_{1:K}) = -\eta \mathcal{L}_q(\mathbf{m}_{1:K})$. With the observation potential added, the potential energy becomes precisely the negative of the log-probability expression that we are trying to optimize (times a constant factor η):

$$V(\mathbf{m}_{1:K}) = V_0(\mathbf{m}_{1:K}) + V^{obs}(\mathbf{m}_{1:K}) \quad (27)$$

$$= -\eta \left[\log p(\mathbf{m}_{1:K}) + \mathcal{L}_q(\mathbf{m}_{1:K})\right] \quad (28)$$

To practically impose the observation potential, we continually introduce external forces into the simulation engine—applying an external force at each point mass equal to the

negative gradient of the observation potential energy at that point, which is given by:

$$\mathbf{f}_k^{obs} = -\nabla_{\mathbf{m}_k} \mathcal{L}(\mathbf{m}_{1:K}). \tag{29}$$

Further, we add a viscous damping force to ensure that the total energy (kinetic plus potential) of the system never increases:

$$\mathbf{f}_k^{damp} = -\gamma \dot{\mathbf{m}}_k. \tag{30}$$

The total energy (kinetic plus potential) of the physical system is guaranteed to converge (since due to dissipation, the energy will decrease if any velocity is nonzero).

The observation forces $\mathbf{f}^{obs}$ are calculated as follows:

$$\mathcal{L}(\mathbf{m}_{1:K}) = \log p(\mathbf{c}_{1:N}|\mathbf{m}_{1:K}) \tag{31}$$

$$= \sum_n \sum_k \alpha_{nk} \log \mathcal{N}(\mathbf{c}_n; \mathbf{m}_k, \Sigma_k) \tag{32}$$

$$= \sum_n \sum_k \alpha_{nk} (\mathbf{c}_n - \mathbf{m}_k)^T \Sigma_k^{-1} (\mathbf{c}_n - \mathbf{m}_k) + \text{const.} \tag{33}$$

Taking the gradient,

$$\mathbf{f}_k^{obs} = \eta \nabla_{\mathbf{m}_k} \mathcal{L}(\mathbf{m}_{1:K}) \tag{34}$$

$$= \eta \sum_n \alpha_{nk} \Sigma_k^{-1} (\mathbf{c}_n - \mathbf{m}_k) \tag{35}$$

In practice, we find that it is satisfactory to ignore the scaling due to Σ_k and simply set

$$\mathbf{f}_k^{obs} = \lambda \sum_n \alpha_{nk} (\mathbf{c}_n - \mathbf{m}_k) \tag{36}$$

for some λ that is chosen empirically to yield fast convergence without instability.

In the present work, the tracked object is modeled as a collection of linked rigid bodies or particles. The physics simulation (we use the Bullet physics engine) uses a linear complementarity problem (LCP) based constraint solver to perform discrete-time integration of the dynamical equations, i.e., to solve for the motion of the objects that is consistent with the Newton-Euler equations of motion. The constraint solver uses a fast iterative algorithm, Projected Gauss-Seidel, to approximately solve the LCP at each timestep. We will not dwell further on the details of the simulation, since our algorithm works with any realistic physics simulation, but the reader is referred to [2] for details.

In summary, the M step involves a hard maximization problem, but our formulation lets us perform the maximization by applying observation forces and time-stepping a generic physics simulator. This scheme allows us to leverage a highly optimized physics simulation with support for a diverse variety of different types of objects.

V. Features

Our tracking algorithm can be straightforwardly generalized to associate a vector of invariant features with each point, not just its (x, y, z) coordinates. Informative features shrink down the set of possible point correspondences, generally enhancing tracking and registration algorithms [18].

Let us first consider the most straightforward addition to the feature vector: color. We used the CIE LAB color space because of its uniformity [7]. With this scheme, each model node and each observed point is associated with a six-dimensional vector (x, y, z, l, a, b); the six-dimensional descriptor vectors are denoted $\mathbf{m}_k^\phi$ and $\mathbf{c}_k^\phi$, respectively. We also choose covariances $\sigma_l, \sigma_a, \sigma_b$ based on estimates of how much these color descriptions vary (e.g. $\sigma_a = \sigma_b = .2,\ \sigma_l = .4$, when $l, a, b \in [0, 1]$). Let Σ_k^ϕ denote the 6×6 covariance matrix $\text{diag}(\sigma_x, \sigma_y, \sigma_z, \sigma_l, \sigma_a, \sigma_b)$.

The modification to the probabilistic model is that the Gaussian conditional likelihood includes the new color coordinates:

$$p(\mathbf{c}_n^\phi | \mathbf{m}_k^\phi, \Sigma_k^\Phi, z_{nk} = 1) = \mathcal{N}(\mathbf{c}_n^\phi; \mathbf{m}_k^\phi, \Sigma_k^\phi). \tag{37}$$

In the inference procedure, this just changes the posterior probabilities $p(z_{kn}|\mathbf{m}_{1:K}^\phi, \mathbf{c}_{1:N}^\phi)$ and thus the expected correspondences α_{kn} that we calculate in the E step. The M step is unchanged: unlike the positions, the nodes' color features are not updated. Conceivably, though, one could update the colors, and this might allow one to texture-map a previously unseen object.

One can also augment the feature vector with a variety of other descriptors of texture.

VI. Real-time algorithm

A real-time implementation of the inference algorithm alternates E step updates with incomplete M step updates. As described in Algorithm 1, we do as many EM iterations as possible before the next observed point cloud arrives.

For each point cloud received:
  $\mathbf{c}_{1:N} \leftarrow$ calculate features from point cloud
  Repeat until new point cloud received:
    $\mathbf{m}_{1:K} \leftarrow$ calculate features from simulation
    $v_{1:K} \leftarrow$ calculate node visibility (14)
    $\alpha_{1:K,1:N} \leftarrow$ calc. expected correspondences (23)
    $\mathbf{f}_{1:K}^{obs}, \mathbf{f}_{1:K}^{damp} \leftarrow$ calculate forces (36)
    Apply forces $\mathbf{f}_{1:K}^{obs}, \mathbf{f}_{1:K}^{damp}$ to nodes
    Step physics simulation

Algorithm 1: Real-time tracking algorithm

VII. Initialization

This section briefly describes how we initialize the model of a previously unseen rope or piece of cloth.

A. Rope

Given a point cloud of a 1D object, such as rope, which may contain some crossings, our initialization procedure is able to infer its 3D curve. The procedure is summarized as follows:

1) Construct a weighted, undirected graph G whose vertices are points in the point cloud. Vertices $\mathbf{p}_i$ and

$\mathbf{p}_j$ with distance $d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$ are connected by an edge of weight d_{ij} if $d_{ij} < thresh$ where, e.g., $thresh = 1\,\mathrm{cm}$.

2) Form the Reeb graph of G based on geodesic distance from an arbitrary point in the graph [9].
3) Consider the set of oriented edges of the Reeb graph. Find the best sequence of oriented edges, according to an objective that favors long paths and penalizes sharp angles and jumps.

B. Surfaces

Given a point-cloud of a 2D object, such as a single piece of cloth, we first generate a triangulated mesh covering the surface. Then we texture-map the surface by projecting the RGB image onto it.

VIII. Software implementation

We have developed a complete pipeline for tracking non-rigid objects in real time. The complete source code is available at the url in the introduction. ROS [15] is used for inter-process communication. The pipeline is illustrated in Figure 4.

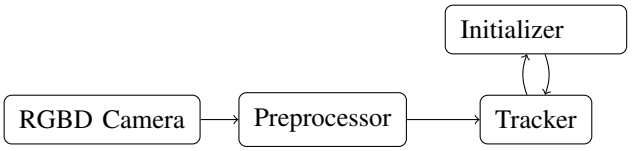

Fig. 4: Tracking pipeline.

First, the sensor (MS Kinect or Asus Xtion Pro Live) acquires an RGB image and a depth image, and the OpenNI driver software projects the 3D points from the depth image onto the RGB camera, generating a colored point cloud.

Next, the preprocessor filters out the points that belong to the object of interest. Since pixel labeling and image segmentation was not our focus, we performed our experiments on top of a green tablecloth, and the human wore green gloves, allowing us to use a simple color-based filtering scheme together with GrabCut [16] (implemented in OpenCV) to separate the object from the its background. In the tracking scenarios involving the robot we also used the kinematic model of the robot to filter out its points. The point cloud is downsampled using a $1\,\mathrm{cm}$ voxel grid before being sent to the tracker. Although the segmentation is not perfect, the tracking algorithm is robust enough to outliers for this to be acceptable.

On the first timestep, the tracker sends the point cloud to the initializer. The initializer first classifies which type of object is present using some shape-based heuristics. Based on the object type, a different initialization method chosen as described in section VII and the object geometry is sent back to the tracker.

The preprocessing and tracking pipeline is able to track rope and towels while robots and and humans are manipulating these objects at moderate speeds. The accuracy and frame-rate is limited by the number of EM iterations it can perform per second, which depends on (1) the density that the point cloud is sampled and (2) the density that nodes are sampled on the surface of the tracked object. See Table I for some typical timing statistics. The bottlenecks in this computation are the correspondence calculation and the physical simulation. It would not be meaningful to include a "frames per second" statistic, since the performance gradually degrades as a function of the speed that the object is moving, but qualitatively speaking, the algorithm can track the objects accurately at 10 frames per second as they are manipulated at a reasonable speed.

	K	N	iter time (ms)
rope	100	≈ 500	20
cloth	1395	≈ 600	50
sponge	150	≈ 200	20

TABLE I: Typical timing results for rope, cloth, and a large sponge. K is the number of object nodes, and N is the number of points in the point cloud associated with the object.

IX. Experimental results

We performed experiments tracking a rope and a cloth using one or two cameras, with or without color information. We colored the rope red, white, and blue so it would contrast with the green background, and we chose an American flag for the same reason. In one set of experiments, the manipulation was performed by a robot; the robot was included in the simulation. In the other set of experiments, the manipulation was performed by a human; of course, the human was not modeled in the simulation, but the tracking still works reliably in the presence of moderate occlusions despite not being aware of the human hand and its contact with the object.

To quantitatively evaluate the perfomance of our tracking algorithm, we collected ground truth data using a commercial motion capture system: the PhaseSpace Impulse. This system precisely tracks the coordinates of a collection of LEDs which are uniquely identified by temporal pulse patterns. We attached 8 LEDs to a rope and 18 LEDs to a piece of cloth for these experiments; the objects are shown in Figure 5.

We enacted the following scenarios to test the performance of our tracking algorithm:

- Six cloth folds by a human: single folds along both diagonals and both principal axes, a double fold into quarters, and a fold into thirds;
- Four rope manipulations by a human: tying an overhand knot, tying a double-overhand knot, tying a figure-eight knot, and untying an overhand knot;
- Three rope manipulations by a robot: tying two figure-eight knots, and tying an overhand knot.

Some camera views from these manipulations are shown in Figure 6.

For the human manipulation scenarios, we ran our tracking algorithm in three data collection modes: (i) with point clouds from one RGBD camera with color features, (ii) with point clouds from two RGBD cameras with color features,

Fig. 5: Ground truth markers on the objects to be tracked.

and (iii) with point clouds from one RGBD camera without color features.

Fig. 6: Manipulation experiments during ground truth data collection with active marker system: human ties a knot (upper left), robot ties a knot (upper right), human folds a towel multiple times (bottom).

The tracking algorithm performed the most robustly in the experiments where a human manipulated cloth. The algorithm tracked the cloth in a qualitatively correct way in all three data collection modes in each of the six tasks. We only observed significant deviations between the state estimate and reality when one region of the cloth occluded another region in the single-camera mode, making the state ambiguous. We measured the mean tracking error by averaging, over all marker observations, the distance from estimated marker position to actual position. (Each marker was associated with a particular point on a triangle in the mesh model, yielding the estimated positions.) The mean error was $2-3\,\mathrm{cm}$ in the cloth manipulation task, as shown in Figure 7.

During the human rope manipulation, the algorithm was also successsful in most trials, i.e., the simulated rope had the correct toplogy at the end of the manipulation, though there were a couple failures that occurred when the person's hand occluded a critical part of the rope, causing an incorrect over- or under-crossing. The errors in this task are shown in Figure 8.

The last type of scenario–robot manipulating rope–was the most challenging and had the lowest success rate. That is because we only used the data from a head-mounted camera, and the rope was mostly occluded by the bulky arms, which made tracking very difficult. The results from this task are shown in Figure 9.

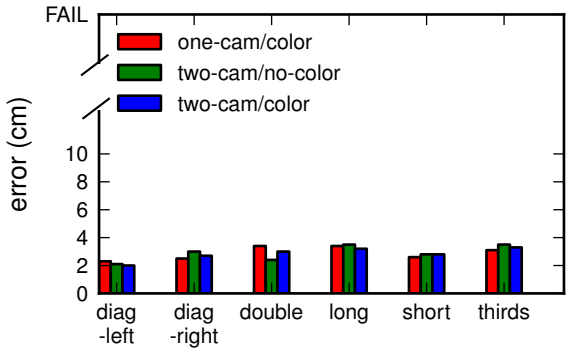

Fig. 7: Mean error in cloth manipulated by human.

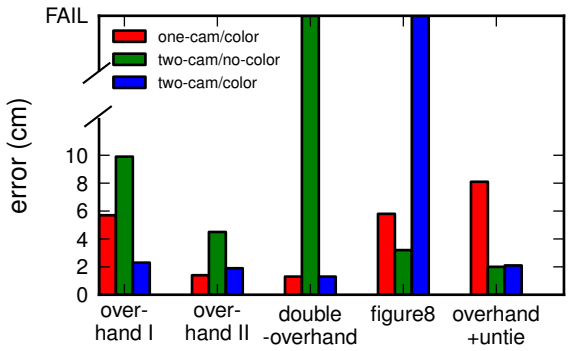

Fig. 8: Mean error in tracking rope manipulated by human. Failures are indicated with maximum-height bars in the plot, indicating that the final configuration is qualitatively different between the estimated and true state (i.e., the knot is not completed). The other bars indicate qualitative success.

Our datasets are posted at the project webpage linked to in the Introduction.

X. Future Work

One limitation of the proposed algorithm is that once the state estimate becomes sufficiently far from reality, it usually does not recover. It would be interesting to augment the tracking approach we proposed to consider multiple hypotheses and escape from local minima in the optimization. Another interesting extension of this work would be to combine tracking with model building to incrementally learn a physical model of a previously unseen object, where the

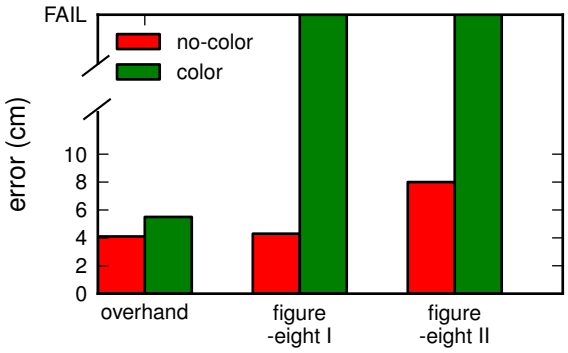

Fig. 9: Mean error in tracking rope manipulated by robot. Failures are indicated with tall bars as above.

model includes the object's geometry, surface texture, and physical properties.

XI. Conclusion

We presented an algorithm for tracking deformable objects, based on point cloud data. The approach is effective for tracking extremely deformable objects, such as rope and cloth, in the presence of occlusions. Unlike previous approaches, our tracking algorithm is built on a generic physics simulation, so it can be used to track essentially anything that one can simulate. Our software implementation can initialize and track a variety of objects in real time using a single algorithm.

We hope that this general-purpose algorithm will enable advances in robotic manipulation by allowing robots to continually track the state of objects they manipulate. Our eventual goal with this line of work is to enable an intelligent robot to maintain an internal, physical simulation of the world that is synchronized with the real world, and to use this representation in its planning and control algorithms.

References

[1] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 1992.

[2] E. Catto. Iterative dynamics with temporal coherence. In *Game Developer Conference*, pages 1–24, 2005.

[3] Yang Chen and Gerard Medioni. Object Modeling by Registration of Multiple Range Images. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, 1991.

[4] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

[5] E. De Aguiar, C. Theobalt, C. Stoll, and H.P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[6] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[7] D.A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[8] D. Hahnel, S. Thrun, and W. Burgard. An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 915–920, 2003.

[9] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM, 2001.

[10] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[11] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, June 1993.

[12] S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4861–4868. IEEE, 2011.

[13] K. Murphy, B. Van Ginneken, J.M. Reinhardt, S. Kabus, K. Ding, X. Deng, K. Cao, K. Du, G.E. Christensen, V. Garcia, et al. Evaluation of registration methods on thoracic ct: The empire10 challenge. *IEEE transactions on medical imaging*, 30(11):1901, 2011.

[14] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 2009.

[16] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[17] M. Salzmann, J. Pilet, S. Ilic, and P. Fua. Surface deformation models for nonrigid 3d shape recovery. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1481–1487, 2007.

[18] G.C. Sharp, S.W. Lee, and D.K. Wehe. Icp registration using invariant features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):90–102, 2002.

[19] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion. *Artificial Intelligence*, 1988.