Tracking Deformable Objects with Normalized Object Coordinate Spaces

Dylan Colli
Robotics Department
University of Michigan
Ann Arbor, Michigan
dfcolli@umich.edu

Yating Lin
Robotics Department
University of Michigan
Ann Arbor, Michigan
yatinlin@umich.edu

Abhinav Kumar
Robotics Department
University of Michigan
Ann Arbor, Michigan
abhin@umich.edu

***Abstract*—To accurately plan and execute tasks involving robotic manipulation of an object, the manipulated object's state must be estimated throughout the task's execution. Given the initial pose and end effector forward kinematics, determining a reasonable estimation of a manipulated rigid body is straightforward. State space estimation in tasks involving the manipulation of deformable objects — such as folding laundry — suffers from the very high dimensional state space that deformable objects exhibit. To address this challenge, we propose a method for tracking deformable objects using learned dynamics and partial point cloud measurements to iteratively update a high density initial point cloud, forming a filtering system. Our method extends previous work in deformable object pose estimation to handle partial, single view observations and to use previous measurements as part of the prediction pipeline. We propose a lightweight dynamics model that utilizes a pre-trained point cloud feature extraction model as well as a measurement update step. We generate trajectory data to train and evaluate the dynamics model and show results comparing our method favorably to a simple tracking baseline.**

***Index Terms*—Tracking, Deformable objects**

I. Introduction

To facilitate accurate planning and execution of task-level manipulation, a reasonable estimate of the manipulated object's state must be obtained. For rigid objects, state estimates can be trivially obtained given a known world-frame translation and rotation. This is due to the low dimensionality of the rigid object's state space and the relatively simple dynamics when compared to deformable objects. Conversely, deformable objects have very high dimensional state spaces and thus, the state estimation of deformable objects poses a significant challenge.

Tracking the pose of deformable objects through time is paramount to enable continuous manipulation of those objects. Real-time control tasks or planning tasks that require replanning will require tracking to provide state estimates at different stages of execution. While methods exist to track deformable objects [1], [2], these methods rely on inflexible perception due to integration of heuristics into the tracking algorithm. These methods rely on specific colors or trackable markers placed on the tracked objects. On the other hand, there exists methods that use deep learning to perceive objects without these markers or hand-engineered heuristics [3]. However, these methods are designed to perform pose detection at a single timestep, and thus do not integrate prior state estimates to perform continuous tracking. We propose an extension to [3] that enables continuous tracking of arbitrary deformable objects within predefined classes.

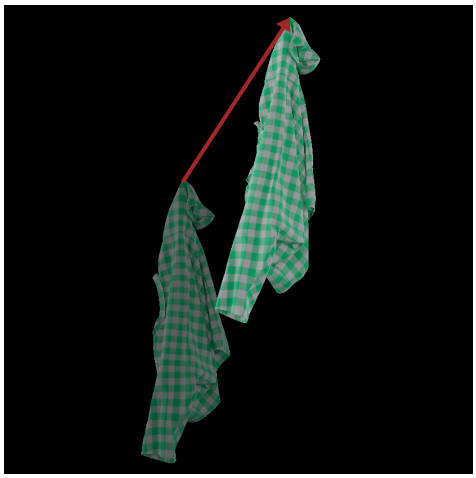

Fig. 1. Overlay of garment initial configuration (faded), final configuration (full color), and gripper motion (red arrow) in a single action of a simulated trajectory used to generate point clouds for training. Note the slight deformation and self-occlusion of the garment's collar induced by gripper motion. Deformation-induced self-occlusion introduces challenges for deformable object tracking methods that are not present in rigid object methods.

Tracking deformable objects is challenging due to their complex dynamics, leading to highly nonlinear evolution of underlying states and visual observations. While the evolution of images or point cloud data is non-linear for rigid objects as well, reasoning about deformable objects greatly increases the space of possible observations and further complicates the evolution of measurements. However, with high-fidelity simulators and deep learning methods that can learn from large amounts of data, we now have techniques that can help us reason about these complex systems. We present a method that can be scaled to leverage large amounts of simulation data to learn the evolution of pose estimates using limited online views.

II. Related Work

GarmentNets [3] is a perception method that enables pose estimation of deformed cloth. It accepts a point cloud generated from fusing multi-angle views of an object and constructs a pose estimate by mapping the point cloud to a Normalized Object Coordinate Space [4], performing shape completion in the canonical space, and estimating the deformation of the object. The main issue preventing GarmentNets from being used for tracking is the multi-angle view it uses for generating pose estimates. While having views of the object from multiple angles allows for more data collection and better initial estimates, a method that can handle single-view predictions for tracking would allow for easier deployment in a real world setting.

Methods such as [1], [2] use statistical methods to track deformable objects. Both methods use expectation maximization techniques to calculate the most likely configuration of points given point cloud measurements. Both methods also assume certain models of the object that they use to structure their estimates. In contrast, methods like GarmentNets do not require these explicit models. We seek to strike a balance between GarmentNets' high degree of expressivity and classical methods' higher degree of online data efficiency.

III. Algorithmic Extension

Our method uses an initial multi-angle point cloud and propagates it forward in time using a dynamics model and corrects the estimate with partial observations. We assume that a full GarmentNets prediction can be made at initialization, which requires views of the object from four angles. Generating four different views at initiailization is trivial and can be done with a static camera by rotating the gripper grasping the object. However, our method does not assume access to those four angles during manipulation where rotating the object to generate four views might be infeasible. Our method works with a single view of the object during manipulation.

Our method has two main components: A dynamics model that predicts the effects of an action on a multi-angle point cloud estimate and a measurement update step that incorporates information from a single-angle point cloud measurement to update the estimate.

The dynamics model that learns an approximation of single-step point cloud dynamics does so using GarmentNets' feature extraction. We learn a function $f(z, u) = \Delta X$, where X is a point cloud, u is an action, and z represent features extracted using the GarmentNets pipeline. We use the PointNet++ [5] feature extractor that is learned as part of GarmentNets as it is trained to reason about point clouds of cloths, allowing us to learn dynamics from less data as we do not have to learn features ourselves. We then input those features, along with an action modeled as a change in gripper position, to a dynamics model which predicts 3D deltas for the positions of each point in the point cloud. Since PointNet++ aggregates information from a neighborhood around each point in the point cloud, we can treat the dynamics model as a pointwise prediction given the pretrained PointNet++ features. We do not predict deltas for the RGB portion of the point cloud in this work. The RGB values are used as inputs to the PointNet++ feature extractor and thus provide potentially useful downstream information, e.g. on shadows induced by garment folds, to the dynamics model. However, we assume they do not undergo a large degree of change during manipulation. This assumption could be relaxed in future work and a good model of the change in the RGB values of the point cloud would be useful for improving the dynamics learning.

The dynamics model is a multi-layer-perceptron with 2 hidden layers with size 512. The input is 131 dimensional, corresponding to 128 dimensions for the PointNet++ features and 3 dimensions for the action. We use ReLU activations and optimize using AdamW [6]. This model has a relatively simple structure that is justified by access to pretrained PointNet++ features.

The loss for this model is the chamfer distance between the predicted point cloud, calculated by adding the predicted position deltas to the input point cloud, and the actual four-view measurement collected when generating the simulated training data, shown in (1). X refers to the ground truth point cloud and $\hat{X}$ refers to the predicted point cloud. We downsample the initial four-view point cloud and the subsequent ground truth measurements to 6000 samples before passing the input through the model and calculating loss to allow for more computationally efficient training. We use the chamfer loss instead of a pointwise L2 loss since we do not have point-to-point correspondences to calculate the L2 loss.

$$J(X, \hat{X}) = .5(\frac{1}{|X|}\sum_{x\in X}\min_{\hat{x}\in\hat{X}}||x-\hat{x}||_2^2+\frac{1}{|\hat{X}|}\sum_{\hat{x}\in\hat{X}}\min_{x\in X}||x-\hat{x}||_2^2) \tag{1}$$

Due to the high quality of the PointNet++ features learned by GarmentNets, training the dynamics model does not require a training time as long as the training time for GarmentNets.

As rolling out the dynamics multiple steps without any sensor updates will result in predictions diverging from the ground truth, we use partial view observations to update the point cloud after computing a forward prediction. This partial view observation can be thought of as viewing the object from a fixed camera. We identify the closest point in the full-view predicted point cloud estimate to each point in the available partial view observation. This defines a partial correspondence between the partial and full views. We optimize a rigid translation between the observed partial view and the corresponding partial view subset of the predicted full-view point cloud to minimize the chamfer distance between these two. The optimized translation distance can be used to correct the predicted full-view point cloud. Then, we directly update the values of the predicted full point cloud for which we have a correspondence to the partial point cloud.

This allows us to directly update part of our estimate with sensor measurements and still use the information from the partial measurement to update the non-corresponding parts of the full point cloud. The combination of applying the dynamics

Algorithm 1 GarmentNets Tracking with Partial-view Point Clouds

function GARMENTNETS(X)
  X_{nocs} = PointNet(X)
  X_{df} = UNet3D(X_{nocs})
  V_{wnf} = WNF(X_{df}, S_1)
  M_{wrf} = IWF(X_{df}, S_2)
  M_{verts}, M_{faces} = MC(V_{wnf})
  return M_{wrf}, M_{faces}
$t = 0$
$\hat{X}_0 = X_0$
while Tracking **do**
  V, F = GarmentNets($\hat{X}_t$)
  $\hat{X}_{t+1}$ = dynamics($\hat{X}_t$, u) #update
  $\hat{X}_{p,t} = \text{knn}(\hat{X}_{t+1}, X_{p,t})$
  $dx = \arg\min_{dx} \text{chamfer\_error}(X_{p,t}, \hat{X}_{p,t} + dx)$
  $\hat{X}_{t+1} = \hat{X}_{t+1} + dx$ #correction
  $\hat{X}_{t+1} = \text{direct\_correspondence\_update}(\hat{X}_{t+1}, X_{p,t})$
  $t = t + 1$

and updating with the measurement can be thought of as following a simple filter's prediction and update steps.

The complete tracking algorithm is presented in Algorithm 1. We define the full-view point cloud measured at initialization as X_0, the current partial-view point cloud measurement as $X_{p,t}$, the full-view prediction as $\hat{X}_t$, and the full-view points corresponding to $X_{p,t}$ as $\hat{X}_{p,t}$. At initialization, we can collect full-view point clouds, but in the subsequent tracking step we are only able to observe a single viewpoint. Therefore, we update the first step full-view point cloud to generate a full-view prediction $\hat{X}_t$ in the subsequent tracking step. Once we have obtained the full-view prediction $\hat{X}_t$, we can then input it into GarmentNets to generate the garment mesh.

In GarmentNets, the transformed point cloud in NOCS Space is represented by X_{nocs}, and prediction dense point cloud features are represented by X_{df}. By performing grid sampling on the dense point cloud features, GarmentNets can use a learned winding number field (WNF) and the marching cubes algorithm (MC) to predict the complete mesh in NOCS space. Then, an output pose of the deformed cloth is predicted using a learned Implicit Warp Field (IWF).

A. Dataset Generation

To generate training and evaluation data for the model, we simulated the resting state and 5 trajectories of 25 T-shirts. We utilized the Blender [7] (version 3.5) Python API to simulate and render both the grasped state that GarmentNets expects as well as the garment trajectories used for this model. Cloth meshes, textures, and physics simulation parameters were obtained via the CLOTH3D dataset [8]. The 25 T-shirts selected for simulation in the training and evaluation of this method were the same CLOTH3D meshes included in the GarmentNets sample dataset available on the public GitHub repository. We obtained the Blender simulation and rendering routines that were used to simulate the grasped states in the original GarmentNets publication and extended them to include simulation of garment trajectories.

The dataset generation pipeline consists of three distinct stages. The first stage involves the simulation of the garment mesh deformation induced by gripper control. The second stage consists of rendering the simulated meshes into 2D RGB and depth images. The third and final stage is the extraction of point clouds from the rendered simulation data.

1) Simulation of Garments: The first stage of the data generation pipeline simulates the mesh deformation induced by gripper control. This stage does not render the images that are necessary for obtaining partial point clouds that are used in training/evaluation. Instead, this stage simulates how the *structure* of the garments deform given gripper motion. This stage can be further broken down into two sub-stages, simulation of the garment's grasped resting state and simulation of the garment motion induced by end effector control.

GarmentNets was trained on garments simulated using meshes from the CLOTH3D dataset. The first sub-stage of garment simulation mirrors exactly the simulation that was done in the original GarmentNets publication. This involved virtually "grasping" a random vertex of the CLOTH3D garment mesh, then simulating the garment's configuration after it was picked up by the gripper and allowed to reach a stable configuration.

The second sub-stage simulates garment deformation induced by gripper control for five separate trajectories. Each trajectory consists of three actions with randomly sampled unit direction vector, $n \in \mathcal{R}^3$, and velocity, $v \in [0.05, 0.20]$ m/s.

2) Rendering Of Simulated Data To 2D Images: Once simulated meshes are obtained, the meshes can be rendered to 2D images that one would expect to receive from a typical RGBD sensor. Blender provides two separate rendering engines for users, Cycles and Eevee. Cycles was built to be a high accuracy, physically based render engine while Eevee is built to optimize render speed. As the accuracy of the depth information is paramount to obtaining high quality point clouds, Cycles was used to render the depth images. Conversely, highly accurate color data is less important to algorithmic performance, so Eevee was used to render the RGB images.

3) Point Cloud Extraction From Rendered Images: The final stage of the data generation pipeline is the extraction of point clouds from the rendered 2D RGBD images. As the camera intrinsics and extrinsics are set by the user, point clouds can be extracted from the 2D images in the typical fashion.

IV. EXPERIMENTS AND RESULTS

A. Experimental Setup

We evaluate our method on simulations of a manipulated garment. These simulations were of a grasped cloth being moved through freespace, with the motion inducing modest garment deformation. We generate evaluation data using 2 different shirts with 5 random trajectories consisting of 75 time steps per shirt for a total of 10 evaluation trajectories or

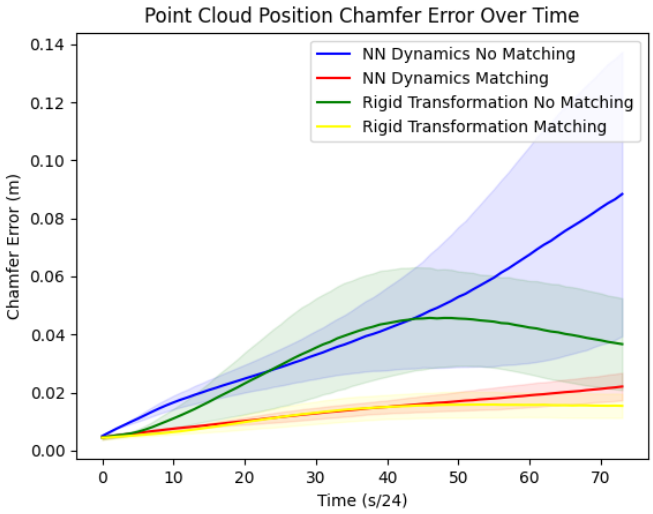

Fig. 2. Chamfer error of point cloud predictions compared to ground truth point clouds over time.

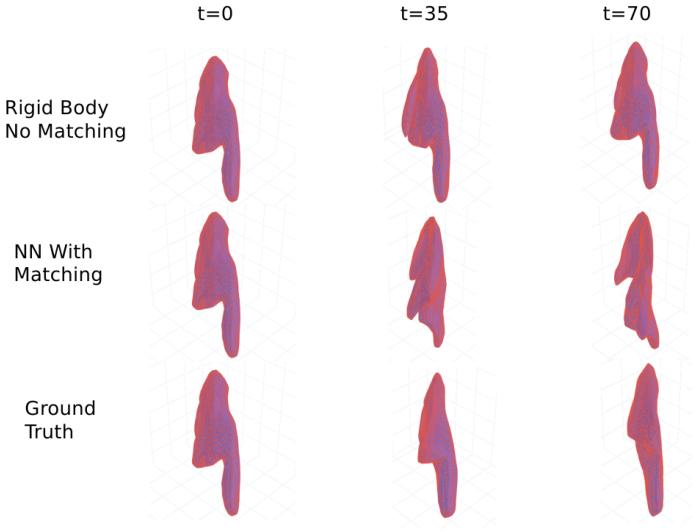

Fig. 3. Comparison of garment mesh tracking using rigid body dynamics without observation matching and MLP dynamics with observation matching at different timesteps. The learned method produces reasonable mesh estimates. Note that the results of the initial timestep correspond to the results of exact inputs that the original GarmentNets method expects (garment grasped and allowed to reach a stable configuration).

750 evaluation time steps. As a baseline, we consider a simple transformation of the cloth's pose calculated by applying the transformation of the gripper to the initial cloth pose. This amounts to a rigid transformation of the garment's initial configuration.

B. Results

We provide error results for four configurations of the filtering system in Fig. 2. The error is quantified using the Chamfer distance between the true four-view point cloud at a given timestep, obtained via simulation, and the predicted four-view point cloud. The error is averaged per timestep over the 10 different evaluation trajectories, with the shaded areas representing 1 standard deviation of the error. These predictions are generated by rolling out single-step dynamics recursively. These results show the utility of the matching method as it clearly attenuates compounding error compared with no matching. As the horizon increases, the error when using the learned dynamics increases faster than the error when using the rigid transformation. This behavior stems from two distinct issues. First, the simulated actions are long enough in duration that the cloth reaches a stable configuration mid-action, causing the rigid deformation with matching to settle to this stable configuration. Second, the model was trained on limited data due to expensive simulation and rendering routines. Though the rigid baseline with matching marginally outperforms the tracking with learned dynamics a majority of the time, the model still outperforms the rigid transformation at certain points of the trajectory (see timesteps 20-40 of 2).

The timesteps with which the learned dynamics prediction outperforms the rigid transformation baseline correspond to the times with which the gripper finishes an action and transitions to a different, randomly sampled action. This change in gripper motion induces high impulse on the garment, causing deformation to ripple through the cloth. As these windows of time exhibit garment deformation, the rigid baseline deviates from true garment configuration. Impressively, the learned model with matching is able to beat the rigid baseline with matching *even after* error accumulation resulting from 20-40 single step dynamics predictions.

With the matching included, the difference between the rigid transformation and the learned transformation shrinks. With more training data and a better tuned model, it may be possible to improve the results of the learned model beyond the rigid transformation, especially if we consider training and evaluation trajectories that induce further garment deformation.

We also present qualitative results that show the GarmentNets predictions when given the predicted future point cloud states in Fig. 3. We present the results from the rigid body transformation without matching as well as the results from using the learned dynamics with matching. These results show reasonable mesh predictions, indicating that the dynamics model does not force point clouds out of distribution of GarmentNets.

V. Conclusion

We demonstrate a method that extends GarmentNets for tracking of deformable objects using a filtering approach of dynamics updates followed by measurement updates from partial point cloud views. The tracking outperforms a rigid baseline and is able to track deformable objects with limited views over time. Future work could include exploration of memory in the tracking, for example using an RNN or transformer model to better enable the model to access views from previous timesteps in combination with the current timestep's view data.

While our method does approximate the underlying dynamics, one area of future work could be in ensuring topological consistency of the updated point cloud. It is possible that the model could predict a new point cloud that is topologically infeasible given the true geometry of the object. By including considerations for this possibility, longer-horizon predictions could become more accurate. In addition, we expect that using L2 loss with known correspondences for learning dynamics would result in a higher quality dynamics model. Downsampling future point clouds in a way that preserves pointwise correspondenes with the current timestep's point cloud would be potential future work.

We believe this method could benefit from training on a larger dataset and for a longer amount of time. Other possible areas of exploration are more sophisticated ways of updating the measurement with the partial view point cloud.

REFERENCES

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

[2] 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.

[3] C. Chi and S. Song, "Garmentnets: Category-level pose estimation for garments via canonical space shape completion," in *The IEEE International Conference on Computer Vision (ICCV)*, 2021.

[4] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2637–2646.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017. [Online]. Available: http://arxiv.org/abs/1706.02413

[6] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017. [Online]. Available: http://arxiv.org/abs/1711.05101

[7] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2023. [Online]. Available: http://www.blender.org

[8] H. Bertiche, M. Madadi, and S. Escalera, "CLOTH3D: Clothed 3D humans," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 344–359.