Model-Driven Feedforward Prediction for Manipulation of Deformable Objects

Yinxiao Li, Yan Wang, Yonghao Yue, Danfei Xu, Michael Case, Shih-Fu Chang, *Fellow, IEEE*, Eitan Grinspun, and Peter K. Allen

***Abstract*— Robotic manipulation of deformable objects is a difficult problem especially because of the complexity of the many different ways an object can deform. Searching such a high-dimensional state space makes it difficult to recognize, track, and manipulate deformable objects. In this paper, we introduce a *predictive, model-driven* approach to address this challenge, using a precomputed, simulated database of deformable object models. Mesh models of common deformable garments are simulated with the garments picked up in multiple different poses under gravity, and stored in a database for fast and efficient retrieval. To validate this approach, we developed a comprehensive pipeline for manipulating clothing as in a typical laundry task. First, the database is used for category and the pose estimation is used for a garment in an arbitrary position. A fully featured 3-D model of the garment is constructed in real time, and volumetric features are then used to obtain the most similar model in the database to predict the object category and pose. Second, the database can significantly benefit the manipulation of deformable objects via nonrigid registration, providing accurate correspondences between the reconstructed object model and the database models. Third, the accurate model simulation can also be used to optimize the trajectories for the manipulation of deformable objects, such as the folding of garments. Extensive experimental results are shown for the above tasks using a variety of different clothings.**

***Note to Practitioners*—This paper provides an open source, extensible, 3-D database for dissemination to the robotics and graphics communities. Model-driven methods are proliferating, and they need to be applied, tested, and validated in real environments. A key idea we have exploited is to have an innovative and novel use of simulation. This database will serve as infrastructure for developing advanced robotic machine learning algorithms. We want to address this machine learning idea ourselves, but we expect the dissemination of the database to other researchers with different agendas and task applications, which will bring wide progress in this area. Our proposed methods, as mentioned earlier, can be easily applied to interrelated areas. One example is that the 3-D shape-based matching algorithm can be used for other objects, such as bottles, papers, and food. After integrating with other robotic systems, the use of the robot can be easily extended to other tasks, such as making food, cleaning room, and fetching objects, to assist our daily life.**

***Index Terms*— Deformable objects, recognition, robotic manipulation, simulation.**

Manuscript received February 27, 2017; revised July 17, 2017; accepted September 10, 2017. This paper was recommended for publication by Associate Editor S. Jeon and Editor H. Ding upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant 1217904. *(Corresponding author: Yinxiao Li.)*

Y. Li, Y. Yue, D. Xu, M. Case, E. Grinspun, and P. K. Allen are with the Department Computer Science, Columbia University, New York, NY 10027 USA (e-mail: yli@cs.columbia.edu).

Y. Wang is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA.

S.-F. Chang is with the Department Computer Science, Columbia University, New York, NY 10027 USA, and also with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2017.2766228

I. Introduction

In THIS paper, we present a feedforward, model-driven approach to address the manipulation of deformable objects, using a precomputed, simulated database of deformable thin-shell object models, where the bending of the mesh models is predominant [9]. The models are detailed, robust, and easy to construct, and using a physics engine, one can accurately predict the behavior of the objects in simulation, which can then be applied to a real physical setting. This paper bridges the gap between the simulation world and the real world. The predictive, feedforward, model-driven approach takes advantages of the simulation and generates a large number of instances for learning approaches, which not only alleviates the burden of data collection, which can be efficiently done in simulation, but also makes adaptation of the methods to other application areas easier and faster. Mesh models of common deformable garments are simulated with the garments picked up in multiple different poses under gravity, and stored in a database for fast and efficient retrieval.

To validate this approach, we developed a comprehensive pipeline for manipulating clothing as in a typical laundry task. First, the database is used to estimate categories and poses of garments in arbitrary positions. A fully featured 3-D volumetric model of the garment is constructed in real time, and volumetric features are then used to obtain the most similar model in the database to predict the object category and pose. Second, the database can significantly benefit the manipulation of deformable objects via nonrigid registration, providing accurate correspondences between the reconstructed object model and the database models. Third, the accurate model simulation can also be used to optimize the trajectories for the manipulation of deformable objects, such as the folding of garments. In addition, the simulation can be easily adapted to new garment models. Extensive experimental results are shown for the above tasks using a variety of different clothings.

Fig. 1 shows a typical pipeline for manipulating clothing as in a laundry task. This paper brings together work addressing all the tasks in the pipeline, which have been previously published in conference papers (see [22] and [24]–[26]).

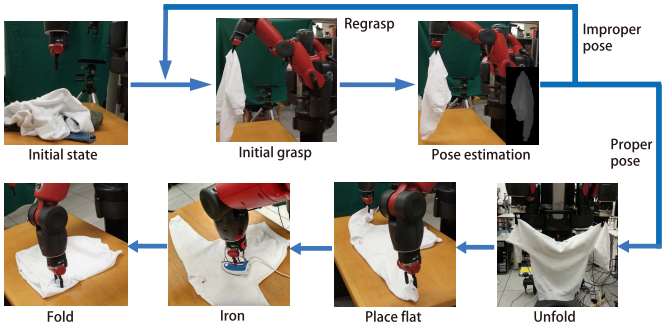


Fig. 1. Overall pipeline of robotic manipulation of deformable objects.

These tasks, with the exception of the ironing task, are all implemented using a feedforward, model-driven methodology, and this paper serves to consolidate all these results into a single integrated whole. This paper has also been extended to include novel garments not found in the database, extended results on regrasping using a much larger data set of objects and examples, quantitative registration results for our hybrid rigid/deformable registration methods, new dense mesh modeling techniques, and a novel dissimilarity metric used to assess folding success. The ironing task is omitted from this paper due to size constraints. Full details on ironing can be found in [23]. In addition, a set of videos of our experimental results are available at: https://youtu.be/fRp05Teua4c.

II. Related Work

A. Recognition

There has been a previous work on the recognition and manipulation of deformable objects. Willimon *et al.* [52], [53] used interactive perception to classify the clothing type. Their work was based on an image-only database of six categories, each of which is with five different items from real garments. Later, they increased the size of the database but still used real garments. Their work focused on small clothing, such as socks and short pants usually consisting of a single color. Cusumano-Towner *et al.* [4], Miller *et al.* [32], Schulman*et al.* [38], and Wang *et al.* [50] have done some impressive works in clothing recognition and manipulation. They have successfully enabled the PR2 robot to fold clothing and towels. Their methods mainly focus on aligning the current edge/shape from observation to an existing shape. A series of works on clothes pose recognition was done in [15]–[17]. They used a simulated database of a single garment with about 18 different grasping points, which were mostly selected on the border when the garment was laid flat. Their work demonstrated the ability to identify clothes pose using registration to prerecorded template images. Doumanoglou *et al.* [5] used a pair of two industrial arms to recognize and manipulate deformable garments. They used a database of depth images captured from 24 real garments, such as a sweater or a pair of pants. Recently, deep learning attracts considerable attention in recognizing clothing category [8] and pose recognition [30], among which convolutional neural network is especially popular in processing the visual information.

With powerful computing resources, reconstructing a 3-D model of the garment and using that to search a precomputed database of simulated garment models in different poses can be more accurate and efficient. With the increasing popularity of the Kinect sensor, there are various methods emerging in computer graphics, such as KinectFusion and its variants [3], [21], [34]. Although these methods have shown success in reconstructing static scenes, they do not fit our scenario directly where a robotic arm is rotating the target garment about a grasping point. Therefore, as mentioned in our previous work [24], we first perform a 3-D segmentation to get the masks of the garment on the depth images, and then invoke KinectFusion for the reconstruction.

Shape matching is another related and long-standing topic in robotics and computer vision. On the 2-D side, various local features have been developed for image matching and recognition [10], [18], [27], which have shown a good performance on textured images. Another direction is shape-context-based recognition [1], [44], [46], which is better for handwriting and character matching. On the 3-D side, Wang *et al.* [49] and Wu *et al.* [54] have proposed the methods to match patches based on the 3-D local features. They extract viewpoint-invariant patches or the distribution of geometry primitives as features, based on which matching is performed. References [7], [36], and [43] apply 3-D shape context as a metric to compute the similarities of the 3-D layout for recognition. However, most of the methods are designed for noise-free human-designed models, without the capability to match the human-designed models against the relatively noisy and incomplete mesh model from Kinect. Our method, published in [22], is inspired by the 3-D shape context [7], but provides the capability of cross-domain matching with a learned distance metric. It also utilizes a volumetric data representation to efficiently extract the features.

B. Manipulation

Reference [37] proposed a method using a dual-arm setup to unfold a garment from pickup. They used a segmented mask to match the prestored template mask to track the states of the garment. The PR2 robot is probably the first robot that has successfully manipulated deformable objects, such as a towel or a T-shirt [29]. The visual recognition, in this paper, targets corner-based features, which does not require a template to match. The subsequent work has improved the prediction of the state of a garment using an hidden Markov model framework by regrasping at the lowest corner point [4]. Reference [5] applied prerecorded depth images to guide the manipulation procedures. Reference [41] used a pair of stereo cameras to analyze the surface of a piece of cloth and performed flattening and unfolding.

One of the applications of our database is to localize the regrasping point during the manipulation by mapping the predetermined points from the simulation mesh to the reconstructed mesh. Therefore, a fast and accurate registration algorithm plays a key role in our method. Rigid or nonrigid surface registration is a fundamental tool to find shape correspondence.

A thorough review can be found in [42]. Our registration algorithm builds on previous rigid and nonrigid registration methods. First, we use iterative closest point (ICP) [2] to rigidly align the garment. We then perform a nonrigid registration to improve matching by locally deforming the garment. Similar to [20], we find the correspondence by minimizing an energy function that describes deformation and fitting.

C. Folding Deformable Objects

With the garment fully spread on the table, attention is turned to understanding its shape and manipulation, such as garment folding. Miller *et al.* [32], [33] have designed a parameterized shape model for unknown garments. A shape model is defined for each type of garment. The goal is to minimize the distance between the observed garment contour points and points from the parameterized shape model. However, the average time for the fitting procedure, reported in this paper, is 30–150 s and sometimes does not converge. The contour-based garment shape model was further improved by Stria *et al.* [39] using polygonal models. The detected garment contour is matched to a polygonal model by removing nonconvex points using dynamic programming. The landmarks on the polygonal model are then mapped to the real garment, and followed by generating a folding plan.

There are other works that have approached the folding garment task with different strategies. Osawa *et al.* [37] used a robot to fold a garment with a special purpose table that contains a plate that can bend and fold the clothes assisted by a dual-arm robot. The robot mainly worked on repositioning the clothes for the plate for each folding action. Within several "flip-fold" operations, the garment can be folded. Another folding method using a PR2 robot was implemented in [48]. The core of their approach was about the geometry reasoning with respect to the cloth model without any physical simulation. Contour fitting at each step took relatively longer than execution of the folding actions, which reduced its efficiency. This was further sped up in [40] using two industrial arms and a polygonal contour model. They showed impressive folding results by utilizing a specifically designed gripper [19] that is suitable for cloth grasping and manipulation.

None of the previous works focus on trajectory optimization for garment folding, which brings uncertainty to the layout given the same folding plan. One possible case is that the garment shifts on the table during one folding action so that the targeted folding position is also moved. Another case is that an improper folding trajectory causes additional deformation of the garment itself, which can accumulate. Our previous work [26] has proved that with effective simulation, bad trajectories can be avoided, and the results of manipulation of the deformable objects are predictable.

III. Problem Formulation

Robotic manipulation of deformable objects is defined as to manipulate the object to a given target shape, given the object in an initial shape [see Fig. 2 (dashed arrow)]. From a quantitative perspective, the problem is to minimize a distance

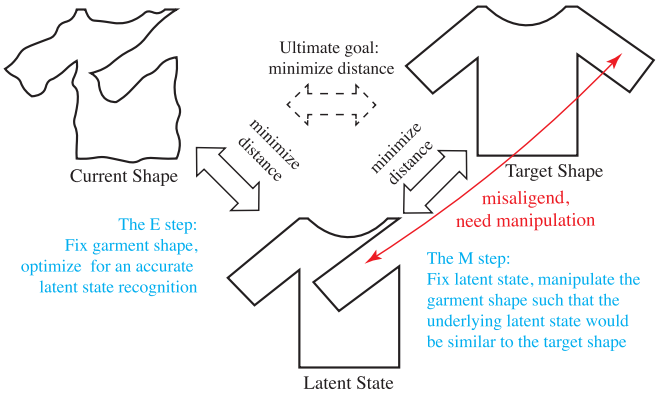


Fig. 2. Problem of deformable object manipulation, shown as the dashed arrow, is very challenging. We introduce a latent state to decouple the two terms and simplify the problem. This naturally leads to a two-stage optimization scheme, illustrated as the blue text.

metric between the current shape S_{current} and the target shape S_{target}

$$\min_{S_{\text{current}}} \|S_{\text{current}} - S_{\text{target}}\|. \tag{1}$$

It is practically challenging, because the sensor noise and the complexity of the many different ways an object can deform. Searching within such a high-dimensional state space makes it difficult to recognize, track, and manipulate deformable objects. To alleviate this problem, we introduce a *latent state* S_{latent} to make the state-space search more tractable, which transforms our problem into

$$\min_{S_{\text{current}}, S_{\text{latent}}} \lambda\|S_{\text{current}} - S_{\text{latent}}\| + (1-\lambda)\|S_{\text{latent}} - S_{\text{target}}\|. \tag{2}$$

Here, λ is a parameter balancing the two terms. This is a common method in mathematics optimization, which simplifies the optimization without moving the actual optimal. In our settings, the latent state S_{latent} corresponds to the "clean" state recognized by the manipulation system, as shown in Fig. 2. This makes the state-space search easier. First, the formulation decouples the two challenges into the sensor noise and the deformation of the object. Second, if we formulate the latent state strategically (e.g., modeling it as the target shape with known deformation), the manipulation planning and the following motion planning will also be simplified. For example, in Fig. 2, with the shown latent state, if we know the correspondences between the latent shape and the target shape, it is straightforward for the system to propose a plan to manipulate the elbow, as shown by the red arrow.

Hence, the introduction of the auxiliary variable S_{latent} decouples S_{current} and S_{target} and naturally leads to a two-stage optimization algorithm similar to the well-known expectation–minimization algorithm. In step one (the E Step), we fix S_{current} and optimize S_{latent} to minimize the objective. And in step two (the M step), we fix S_{latent} and optimize S_{current} to minimize the objective. The process is also illustrated as the blue texts in Fig. 2. From a robotics perspective, the E step is recognition, and the M step is manipulation. Therefore, we derive an iterative framework, which does recognition and manipulation iteratively from a mathematics perspective.

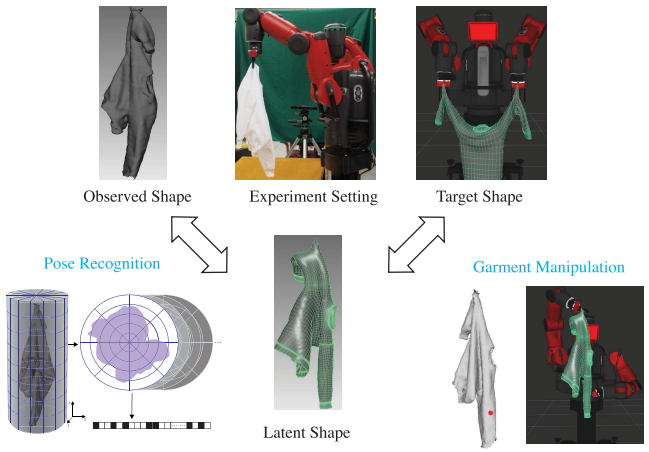


Fig. 3. Proposed pipeline using deformable garment manipulation as an example.

More specifically, we use a set of predefined poses of the object as the *anchor* of the latent state. Taking a garment as an example, each pose can be represented by a set of images or a deformed 3-D model from them. Physically, having a robot arm to pick up an object and capture its appearance is too slow and cannot span the large space we hope to learn. Given the physical nature of this training set, it can be very time-consuming to create, and may have problems encompassing a wide range of garments and different fabrics, which we can more easily accommodate in the simulation environment. Therefore, we use a physics engine to simulate a set of 3-D shapes of the garment under gravity (*anchor* shapes), with different points grasped by the robotic arm. We assume that the observed shape in the manipulation process can be derived from one of those anchor shapes with slight nonrigid deformation. The benefit of this design is twofold. For the recognition step, we further decompose the rigid transform estimation from the nonrigid registration. And with the help of a large number of presimulated 3-D shapes, we can achieve a tradeoff between accuracy and efficiency. For the manipulation step, such a design allows us to know pointwise correspondences between the latent and target shape, so the motion planning can be performed easily.

In Sections IV–VII, we will use garment folding as an example. Fig. 3 shows the experimental settings for our algorithm: a Baxter robot with a Kinect sensor grasps a garment and predicts the grasping location. It then uses its robotic arms to manipulate the garment to a predefined pose. This procedure will be performed iteratively until the observed shape is sufficiently close to the target. Specifically, three key factors of the system will be discussed in the following sections. First, in Section IV, we will introduce how the *anchor* poses are generated. Then, in Section V, the formulation and the optimization of the registration cost are discussed, with the final goal to recognize the pose of the garment as well as pointwise correspondences between the anchor shape and the observed shape. Note we can easily obtain pointwise correspondences between the target shape and anchor shapes given they are both from physics simulation. In Section VI, we further explain how to manipulate the object given a latent shape and a target shape, using Bézier curves as the optimized trajectories.

We want to point out that some of the models used involve a training stage. Specifically, the E step requires distance learning and feature extraction before actual optimization. The registration, the M step, and the folding step do not require any training, and thus, the optimization step can be performed directly. Therefore, the entire recognition and manipulation process can be done after the training stage of the E step, which will be described in detail in Section V-C.

IV. Database for Deformable Object Recognition

As mentioned in Section III, a core component of both the recognition step and the manipulation step is an off-line simulated database. In recognition, instead of explicitly modeling how the observed shape was deformed from the target shape, we can use a precalculated database to handle all the possible ways of deformation, and transform the recognition problem as a shape retrieval problem. In manipulation, since we know how each simulated model was deformed from the target shape, motion planning will also be much less challenging on such clean input. The idea of handling deformation by simulation essentially trades off efficiency with accuracy. The computation burden is moved from the online stage to the off-line stage, so we can use sophisticated algorithms to get realistic and accurate shapes after deformation. But on the other hand, because we cannot cover every possible grasping points and materials during the simulation, it also compromises some accuracy. To address this problem, we further introduce a nonrigid registration method in addition to the rigid registrations between the database models and the observed models in Section V. From another perspective, this is also a good example of using prior knowledge to guide robots to follow the steps of a task.

More specifically, we have developed an off-line simulation pipeline whose results are good enough to support various applications, using advanced simulators, such as *Maya* [11] to simulate deformable objects. In this way, we can produce thousands of exemplars efficiently, which can be used as a corpus for learning the visual appearances of the deformed garments. The off-line simulation is time efficient, noisefree, and more accurate compared with acquiring data via sensors from real objects. Simulation models do not suffer from occlusion or noise as compared with physically scanned models. In the off-line simulation, we use a few well-defined garment mesh models, such as sweaters, jeans, and short pants. Similar garment mesh models can be obtained from Poserworld [12] and Turbo Squid [13]. We can also generate models by using our own "*Sensitive Couture*" software [47]. Fig. 4 shows a few of our current garment models rendered in *Maya* software.

For each grasping point, we compute the garment layout by hanging under gravity in the simulator. In *Maya*, a mesh model can be converted into an *nCloth* model, which can be then simulated with some cloth properties, such as hanging and falling down. *Maya* also allows for the control of cloth thickness and deformation resistance. In addition, any vertex

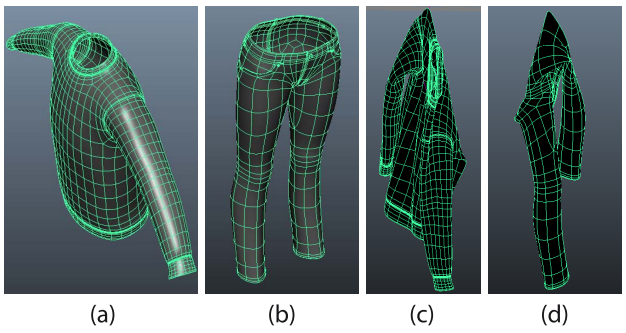


Fig. 4. (a) and (b) Original garment mesh models of a sweater and a pair of jeans rendered in *Maya*. (c) and (d) Simulation result of hanging (a) and (b) under gravity, respectively.

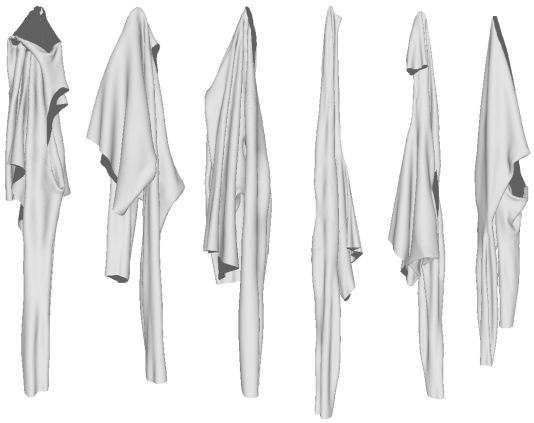


Fig. 5. Six different mesh models of the same sweater, but picked up on different points, simulated in *Maya*.

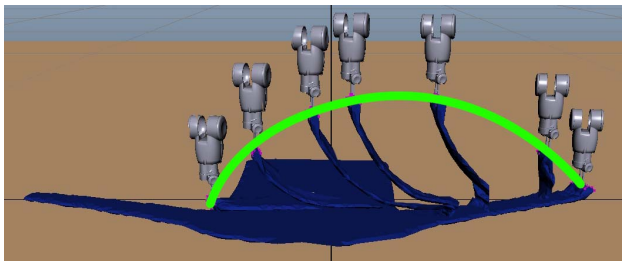


Fig. 6. Folding trajectory for a sweater simulated in *Maya*.

on the mesh can be selected as a constraint point to simulate a draping effect. The hanging under gravity effect of the garment models is shown in Fig. 4. Fig. 5 shows a small sample of different picking points of a single garment hanging under gravity that simulated in *Maya*. We release the simulation script in http://www.cs.columbia.edu/ỹli/garment_db/ so that researchers can produce training exemplars using their own models. Fig. 6 shows a simulated trajectory of the garment manipulation.

V. Pose Recognition

With the off-line simulated database, we solve the pose recognition problem in three steps. First, we use a Kinect sensor to capture many depth images from different viewpoints of the garment by rotating it as it hangs from a robotic arm. We then reconstruct a smooth 3-D model from the depth input, extract compact 3-D features from it, and finally match against the off-line model database to recognize its pose. Finally, we do a nonrigid registration to find more accurate pointwise correspondences between the observed 3-D model and the database model. The overall pipeline for the pose recognition is shown in Fig. 7. We propose a binary feature to do efficient shape retrieval. The training process of the feature and the associated weight is shown in Fig. 7 (red box).

Viewing the recognition step from a big picture, instead of solving the original problem

$$\min_{S_{\text{latent}}} \|S_{\text{current}} - S_{\text{latent}}\|. \tag{3}$$

Our initial design did not contain the binary 3-D feature and the fast matching design. But then, we found that the pose recognition part is the bottleneck of the system in terms of speed. Therefore, we introduce two assumptions to simplify the problem. First, with the use of the presimulated database, we assume that the latent shape can only be an instance from the database. Second, we introduce an efficient binary feature to describe the shape, and thus define the norm $\|\cdot\|$ as a weighted Hamming distance. With sufficient samples in the database, and a proper definition of the norm using a large-margin learning-to-rank schema, these two assumptions can significantly improve the recognition speed, without introducing much error. We will introduce each component in Sections V-A–V-D.

A. 3-D Reconstruction

Given the above-described model database, we now need to generate depth images and match against the database. Direct recognition from depth images suffers from the problems of self-occlusion and sensor noise. This naturally leads to our new method of first building a smooth 3-D model from the noisy input, and then performing recognition in 3-D. However, how to do such reconstruction is still an open problem, although there exist approaches of obtaining high-quality models from noisy depth inputs, such as KinectFusion [34], which requires the scene to be static. In our data collection settings, the target garment is being rotated by a robotic arm, which invalidates the KinectFusion's assumptions. We solve this problem by first segmenting out the garment from its background, and then invoke KinectFusion to obtain a smooth 3-D model, assuming that the rotation is slow and steady enough, such that the garment will not deform in the process.

Segmentation: Before diving into the reconstruction algorithm, let us first define some notation. Given the intrinsic matrix F_d of the depth camera and the ith depth image I_i, we are able to compute the 3-D coordinates of all the pixels in the camera coordinate system with $[x_{ci}\ y_{ci}\ z_{ci}]^T = F^{-1} d_i [u_i\ v_i\ 1]^T$, in which (u_i, v_i) is the coordinate of a pixel in I_i, with d_i as the corresponding depth, and (x_{ci}, y_{ci}, z_{ci}) is the corresponding 3-D coordinate in the camera coordinate system.

Our segmentation is then performed in the 3-D space. We ask the user to specify a 2-D bounding box on the

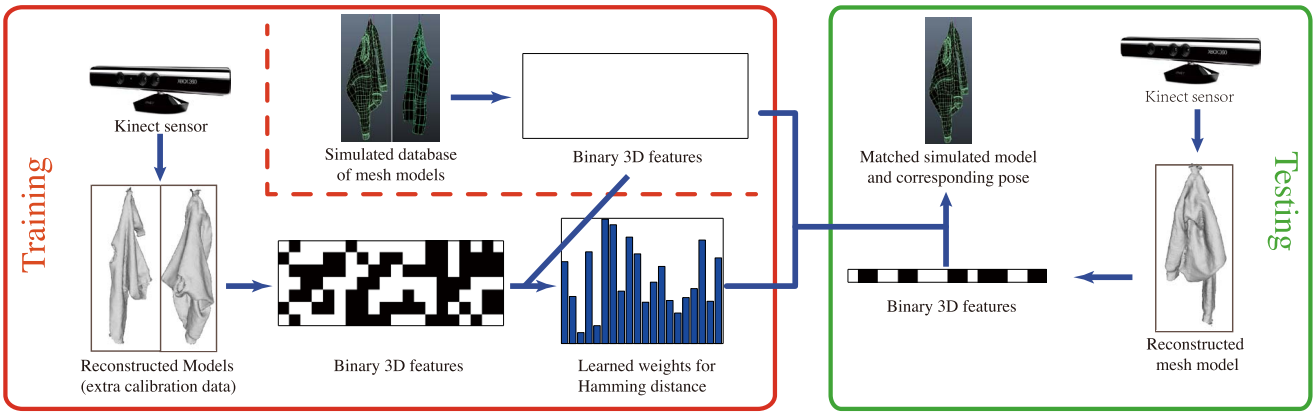


Fig. 7. Overview of the pipeline for the pose recognition step. In the off-line training stage (the red rectangle), we extract a tailored binary feature from the simulated database, and learn a weighted Hamming distance from additional calibrated data collected from the Kinect. In the online testing stage (the green rectangle), we reconstruct a 3-D model from the depth input, find the NN from the simulated database with the learned distance metric, and then adopt the pose of the matched model as the output.

depth image $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ with a rough estimation of the depth of the garment $(z_{\min}, z_{\max})$. Given that the data collection environment is reasonably constrained, we find even one predefined bounding box works well for all categories of garments. The box needs to be redrawn only when dramatic changes of the environment happen, such as when the sensor or the robot is moved. Then, we adopt all the pixels having their 3-D coordinates within the bounding box as the foreground, resulting in a series of masked depth images $\{I_i\}$ and their corresponding 3-D points, which will be fed into the reconstruction module.

The 3-D reconstruction is done by feeding the masked depth images $\{I_i\}$ into KinectFusion, while the unrelated surroundings are eliminated, leaving the scene to reconstruct as static. This process can be done in real time. In addition to a smooth mesh, the KinectFusion library also generates a signed distance function (SDF) mapping, which will be used for 3-D feature extraction. The SDF is defined on any 3-D point (x, y, z). It has the property that it is negative when the point is within the surface of the scanned object, positive when the point is outside a surface, and zero when it is on the surface. We will use this function to efficiently compute our 3-D features in Section V-B.

B. Feature Extraction

Inspired by 3-D shape context [1], we design a binary feature to describe the 3-D models. In our method, the features are defined on a cylindrical coordinate system fit to the hanging garment as opposed to traditional 3-D shape context, which uses a spherical coordinate system [7].

The cylinder is divided into N layers with the same height. And for each layer, as shown in Fig. 8 (top-right), we *uniformly* divide the world space into (R rings) $\times$ (Φ sectors) in a polar coordinate system, with the largest ring covering the largest radius among all the layers. The center of the polar coordinate system is determined as the mean of all the points in the highest layer, which usually contains the robot gripper. Note we do a uniform division instead of logarithm division of r as shape context does. The reason why shape context uses

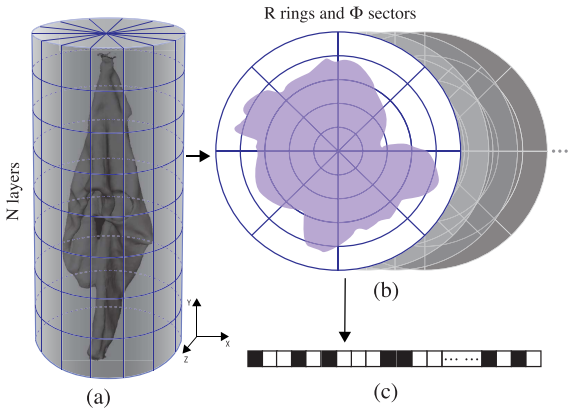


Fig. 8. Feature extraction from a reconstructed mesh model. (a) Bounding cylinder of a garment is cut into several layers. (b) Set of layers (sections). For each layer, we divide it into cells via rings and sectors. (c) Binary feature vector collected from each cell. Details are described in Section V-B.

the logarithm division of r is that the cells farther from the center are less important, which is not the case in our settings. For each layer, instead of doing a point count as in the original shape context method, we check the SDF of the voxel which the center of the polar cell belongs to, and fill one (1) in the cell if the SDF is zero or negative (i.e., the cell is inside the voxel), otherwise zero (0). Finally, all the binary numbers in each cell are collected in an order (e.g., with ϕ increasing and then r increasing), and are concatenated as the final feature vector.

The insight behind is, to improve the robustness against local surface disturbance due to friction, that we include the 3-D voxels *inside* the surface in the features. Note we do not need to do the time-consuming classification (e.g., ray tracing) to determine whether each cell is inside the surface, but only need to look up their SDFs, thus dramatically speed up the feature extraction.

Matching Scheme: Similar to shape context [1], when matching against two shapes, we conceptually rotate one of them and adopt the minimum distance as the matching cost to

provide rotation invariance. That is

$$\text{Distance}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \min_i \|R_i \boldsymbol{x}_1 \oplus \boldsymbol{x}_2\|_1 \tag{4}$$

in which $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{B}^{\Phi RN}$ are the features to be matched ($\mathbb{B}$ is the binary set $\{0, 1\}$), $\oplus$ is the binary XOR operation, and R_i is the transform matrix to rotate the feature of each layer by $2\pi/\Phi$. Recall that both features to be matched are compact binary codes. Thus, such conceptual rotation as well as Hamming distance computation can be efficiently implemented by integer shifting and XOR operations, resulting in matching that is even faster than the Euclidean distance given reasonable Φ values (e.g., $\Phi = 10$).

C. Domain Adaptation

Now, we have a feature vector representation for each model in the simulated database and for the query. A natural idea is to find the nearest neighbor (NN) of the query in the database and transfer the metadata, such as category and pose from the NN to the query. But a naive NN algorithm with the Euclidean distance does not work here, because even for the same garment and the same grasping point by the robot, the way it deforms may still be slightly different due to friction. This requires a solution in the matching stage, especially given that it is impractical to simulate every object with all the possible materials. Therefore, essentially, we are doing cross-domain retrieval, which generally requires a "calibration" step to adapt the knowledge from one domain (simulated models) to another (reconstructed models).

1) Weighted Hamming Distance: Similar to the distance calibration in [51], we use a *learned* distance metric to improve the NN accuracy, that is

$$\text{BestMatch}_{\boldsymbol{w}}(\boldsymbol{q}) = \arg\min_i \boldsymbol{w}^T \left(\hat{\boldsymbol{x}}_i \oplus \boldsymbol{q}\right) \tag{5}$$

in which $\boldsymbol{q}$ is the feature vector of the query, i is the index of models in the simulated database, and $\oplus$ is the binary XOR operation. $\hat{\boldsymbol{x}}_i = \hat{R}_i \boldsymbol{x}_i$ indicates the feature vector of the ith model, with $\hat{R}_i$ as the optimal R in (4).

The insight here is we aim to grant our distance metric more robustness against material properties by assigning larger weights to the regions invariant to the materials (this amplifies the features that are more intrinsic for the recognition task).

2) Distance Metric Learning: To learn the weighted Hamming distance, we use an extra set of mesh models collected from a Kinect using the same setting in Section V-A as *calibration data*. Only a small amount of data is needed for each category (e.g., 5 models in 19 poses for sweater model). To determine the weight vector $\boldsymbol{w}$, we then formulate the learning process as an optimization problem of minimizing the empirical error with a large-margin regularizer

$$\begin{aligned} &\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_j \xi_j \\ &\text{s.t. } \boldsymbol{w}^T \left(\hat{\boldsymbol{x}}_i \oplus \boldsymbol{q}_j\right) < \boldsymbol{w}^T \left(\hat{\boldsymbol{x}}_k \oplus \boldsymbol{q}_j\right) + \xi_j \\ &\quad \forall j, \quad \forall y_i = l_j, y_k \neq l_j \\ &\quad \xi_j \geq 0 \end{aligned} \tag{6}$$

in which $\hat{x}_i$ is the orientation-calibrated feature of the ith model (from the database), with y_i as the corresponding ground truth label (i.e., the index of the pose). $\boldsymbol{q}_j$ is the extracted feature of the jth training model (from Kinect), with l_i as the ground truth label. We wish to minimize $\sum_i \xi_i$, which indicates how many wrong results the learned metric $\boldsymbol{w}$ give, with a quadratic regularizer. C controls how much penalty is given to wrong predictions.

This is a nonconvex and even nondifferentiable problem. Therefore, we employ the RankSVM [14] to obtain an approximate solution using the cutting-plane method.

3) Knowledge Transfer: Given the learned $\boldsymbol{w}$, in the testing stage, we then use (5) to obtain the NN of the query model. We directly adopt the grasping point of the NN, which is known from the simulation process, as the final prediction. We will describe experiments using the method in Section VII-A.

D. Deformable Registration

After obtaining the location of the current grasp point, we seek to register the reconstructed 3-D model to the ground truth garment mesh to establish point correspondences. The input to the registration is a canonical reference ("source") triangle mesh S_{latent} that has been computed in advance and stored in the garment database, and a target triangle mesh S_{current} representing the geometry of the garment grasped by the robot, as acquired by 3-D scans of the grasped garment.

The registration proceeds in three steps. First, we scale the source mesh S_{latent} to match its size to the target mesh S_{current}. Next, we apply an ICP technique to rigidly transform the source mesh S_{latent} (i.e., via only translation and rotation). Finally, we apply a nonrigid registration technique to locally deform the source mesh S_{latent} toward the observation S_{current}. In Section V-D1, following the standard practice of 3-D registration, we use "source model" to refer to S_{latent} and "target model" to refer to S_{current}.

1) Scaling: First, we compute a representative size for each of the source and target meshes. For a given mesh, let a_i and $\mathbf{g}_i$ be the area and barycenter of the ith triangle. Then, the area-weighted center $\mathbf{c}$ of the mesh is

$$\mathbf{c} = \sum_i^{N_S} a_i \mathbf{g}_i \Big/ \sum_i^{N_S} a_i \tag{7}$$

where N_S is the number of vertices of the source mesh S. Given the area-weighted center, the representative size l of the mesh is given by

$$l = \sum_i^{N_S} a_i \|\mathbf{g}_i - \mathbf{c}\| \Big/ \sum_i^{N_S} a_i. \tag{8}$$

Let the representative sizes of the source and target meshes be l_S and l_T, respectively. Then, we rescale the source mesh by a factor of l_T/l_S, which is a 3-D vector in the x, y, and z space with respect to a 3-D reference point.

2) Computing the Rigid Transformation: We use a variant of ICP [2] to compute the rigid transformation. ICP iteratively updates a rigid transformation by: a) finding the closest

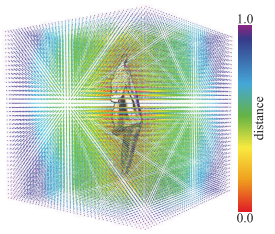


Fig. 9. Visualization of distance function given a mesh. Right: color bar shows the normalization distance.

point $\mathbf{w}_j$ on the target mesh T for each vertice $\mathbf{v}_j$ of the source mesh S; b) computing the optimal rigid motion (rotation and translation) that minimizes the distance between $\mathbf{w}_j$ and $\mathbf{v}_j$; and c) updating the vertices $\mathbf{v}_j$ via this rigid motion.

To accelerate the closest point query, we prepare a grid data structure during preprocessing. For each grid point, we compute the closest point on the target mesh once using fast sweeping [45], and store for runtime using both the found point and its distance to the grid point, as shown in Fig. 9, to avoid multiple queries and computations.

At runtime, we approximate the closest point query for vertex $\mathbf{v}_j$ by searching only among those eight precomputed closest points corresponding to the eight grid points surrounding $\mathbf{v}_j$, thereby reducing the complexity of the closest point query to $O(1)$ per vertex. After establishing point correspondences, we compute the optimal rotation and translation for registering $\mathbf{v}_j$ with $\mathbf{w}_j$ [2]. We iteratively compute point correspondences and rigid motions until successive iterations converge to a fixed rigid motion, yielding a rigidly registrated source mesh $\bar{S}$.

3) Nonrigid Registration: Given a candidate source mesh $\bar{S}$ obtained via rigid registration, our nonrigid registration seeks the vertex positions $\mathbf{v}_j$ of the source mesh S that minimize

$$E_{\bar{S},T}(S) = E_{\text{fit}}(S, T) + E_{\text{def}}(S, \bar{S}) \tag{9}$$

where $E_{\text{fit}}(S, T)$ penalizes discrepancies between the source and target meshes and $E_{\text{def}}(S, \bar{S})$ seeks to limit and regularize the deformation of the source mesh away from its rigidly registrated counterpart $\bar{S}$. The term

$$E_{\text{fit}} = \sum_{i=1}^{N_S} (\text{dist}(\mathbf{g}_i))^2 \bar{A}_i \tag{10}$$

penalizes deviation of the target mesh. Here, $\mathbf{g}_i$ is the barycenter of the triangle i and $\text{dist}(\mathbf{g}_i)$ is the distance from $\mathbf{g}_i$ to the closest point on the target mesh. As in the rigid case, we use the precomputed distance field to query for the distance.

It might appear that the fitting energy E_{fit} could be trivially minimized by moving each vertex of mesh S to lie on mesh T. In practice, however, this does not work, because all of the geometry of the precomputed reference mesh $\bar{S}$ is discarded; instead, the geometry of this mesh, which was precomputed using fabric simulation, should serve as a prior. Thus, we introduce a second term to retain as much as possible the geometry of the reference mesh $\bar{S}$.

The deformation term $E_{\text{def}}(S, \bar{S})$, derived from a physically based energy (see [9]), is a sum of three terms

$$E_{\text{def}}(S, \bar{S}) = \kappa E_{\text{area}} + \beta E_{\text{angle}} + \alpha E_{\text{hinge}} \tag{11}$$

where α, β, and κ are user-specified coefficients. The term

$$E_{\text{area}} = \sum_{i=1}^{N_S} \frac{1}{2} \left(\frac{A_i}{\bar{A}_i} - 1 \right)^2 \bar{A}_i \tag{12}$$

penalizes changes to the area of each mesh triangle. Here, A_i is the area of the triangle i and $\bar{\cdot}$ refers to a corresponding quantity from the undeformed mesh $\bar{S}$. The term

$$E_{\text{angle}} = \sum_{i=1}^{N_S} \sum_{k=1}^{3} \frac{1}{6} \left(\frac{\theta_{ik}}{\bar{\theta}_{ik}} - 1 \right)^2 \bar{A}_i \tag{13}$$

penalizes the shearing of each mesh triangle, where θ_{ik} is the kth angle of the triangle i. The term E_{hinge} [9]

$$E_{\text{hinge}} = \sum_{e} (\theta_e - \bar{\theta}_e)^2 \|\bar{e}\| / \bar{h}_e \tag{14}$$

penalizes bending, measured by the angle formed by adjacent triangles. Here, θ_e is the *hinge angle* of edge e, i.e., the angle formed by the normals of the two triangles incident to e; $\|\bar{e}\|$ is the length of the edge e, and $\bar{h}_e$ is a third of the sum of the heights of the two triangles incident to the edge e.

We used the secant-version of the Levenberg–Marquardt (L–M) method [28] to seek the source mesh S that minimizes the energy (9). Sample registration results are shown in Fig. 14. In Section VII-B, we describe registration and regrasping experiments using the method described in this section.

VI. Garment Manipulation

A. Folding by Minimizing Distance

With the garment lay flat on the table, it can be manipulated by minimizing the distance between the *current state* and the *desired state*. Taking garment folding as an example, the goal is specified by the initial and folded shapes of the garment and by the starting and target positions of the grasp point (as in Fig. 10). The key factor here is to compute optimal folding trajectories that can minimize the distance between the folded shape (current state) and the desired folded shape (desired state). Practically, it is difficult to optimize trajectories for a desired folded shape. Therefore, we seek the trajectory that effects the desired set of folds in the simulation. With the comparable simulation environment to the real world, the simulation results, e.g., optimized trajectories, can be transferred to a real robot.

We use a Bézier curve [6] to describe the trajectory. An nth order Bézier curve $\mathbf{T}(u)$ has $(n + 1)$ control points $\mathbf{P}_k = (P_{k,x}, P_{k,y}, P_{k,z})^T \in \mathbb{R}^3$, defined by

$$\mathbf{T}(u) = \sum_{k=0}^{n} B_k^n(u) \mathbf{P}_k \tag{15}$$

where $B_k^n(u) = \binom{n}{k} (1-u)^{n-k} u^k$ (Bernstein basis).

We use $n = 3$ for simplicity, but our method can be easily extended to deal with higher order curves. $\mathbf{P}_0$ and $\mathbf{P}_3$ are fixed to the specified starting and target positions of the grasp point (as in Fig. 10). The intermediate control points $\mathbf{x} = (\mathbf{P}_1^T, \mathbf{P}_2^T)^T$

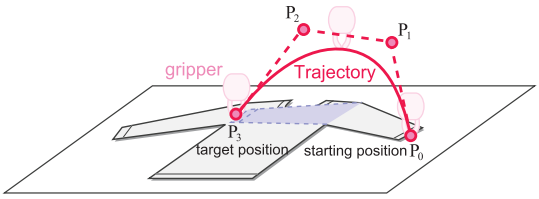


Fig. 10. Example of the folding task: we want to fold a sleeve into the blue target position, by using a robotic gripper to move the tip of the sleeve (grasp point) from the starting position ($\mathbf{P}_0$) to the target position ($\mathbf{P}_3$), following a trajectory, shown as the red curve. $\mathbf{P}_1$ and $\mathbf{P}_2$ are knot points that form the Bézier trapezoid.

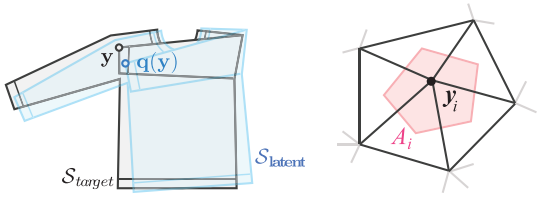


Fig. 11. Left: distance captures the misalignment between S_{target} and S_{latent} by integrating the distance between the corresponding points $\mathbf{y} \in S_{\text{target}}$ and $\mathbf{q}(\mathbf{y}) \in S_{\text{latent}}$ over the garment. Right: barycentric dual area A_i associated with this vertex $\mathbf{y}_i$ is defined as the area of the polygon created by connecting the barycenters of the triangles adjacent to $\mathbf{y}_i$.

can then be adjusted to define a new trajectory using the objective function defined as follows:

$$\mathbf{x}_{\text{opt}} = \underset{\mathbf{x}}{\text{argmin}} \underbrace{\{l_{\mathbf{x}} + \alpha \| S_{\text{latent}} - S_{\text{target}} \|\}}_{C(\mathbf{x})}^2. \tag{16}$$

Here, $C(\mathbf{x})$ is a cost function with two terms. The first term penalizes the trajectory length $l_{\mathbf{x}}$, thus preferring a folding path that is efficient in time and energy. The second term seeks the desired fold, by penalizing distance $\| S_{\text{latent}} - S_{\text{target}} \|$ between the desired folded shape S_{target}, compared with the shape S_{latent} obtained by the candidate folding trajectory $\mathbf{x}$, as predicted by a cloth simulation; we used a physical simulation engine [31], for the cloth simulation. The weight α balances the two terms; we used $\alpha = 10^3$ in our experiment.

Intuitively, the distance measures the difference between the desired folded shape and the folded garment in simulation. We define the distance term as

$$\| S_{\text{latent}} - S_{\text{target}} \| = \frac{1}{|S_{\text{target}}|} \int_{S_{\text{target}}} \| \mathbf{q}(\mathbf{y}) - \mathbf{y} \| dA \tag{17}$$

where $|S_{\text{target}}|$ is the total surface area of the garment mesh including both sides of the garment, $\mathbf{y} \in S_{\text{target}}$ is a point on the target folded shape S_{target}, $\mathbf{q}(\mathbf{y}) \in S_{\text{target}}$ is the corresponding point on the simulated folded shape, and dA is the area measure [see Fig. 11 (left)]. Our implementation assumes that S_{target} and S_{latent} are given as triangle meshes, and discretizes (17) as

$$\| S_{\text{latent}} - S_{\text{target}} \|_d = \frac{1}{|S_{\text{target}}|} \sum_i \| \mathbf{q}_i - \mathbf{y}_i \| A_i \tag{18}$$

where $\mathbf{y}_i$ is the barycenter of the ith triangle on the target shape, $\mathbf{q}_i$ is the (corresponding) barycenter of the ith triangle on the simulated shape, and A_i is the barycentric area of the ith triangle on the target shape, as defined in Fig. 11 (right).

To compute the trajectory length $l_{\mathbf{x}}$, we use De Casteljau's algorithm [6] to recursively subdivide the Bézier curve $\mathbf{T}$ into a set of Bézier curves $\mathbf{T}^{(j)}$, until the deviation between the chord length ($\|\mathbf{P}_0^{(j)} - \mathbf{P}_3^{(j)}\|$) and the total length between the control points ($\sum_{i=0}^{2} \|\mathbf{P}_i^{(j)} - \mathbf{P}_{i+1}^{(j)}\|$) for each subdivided curve $\mathbf{T}^{(j)}$ is sufficiently small. Then, $l_{\mathbf{x}}$ is approximated by summing up the chord lengths of all the subdivided curves: $l_{\mathbf{x}} \approx \sum_j \|\mathbf{P}_0^{(j)} - \mathbf{P}_3^{(j)}\|$.

We initialize $\mathbf{P}_1$ and $\mathbf{P}_2$ as

$$\mathbf{P}_1 = \frac{2}{3}\mathbf{P}_0 + \frac{1}{3}\mathbf{P}_3 + h\|\mathbf{P}_0 - \mathbf{P}_3\|\mathbf{e}_v \tag{19}$$

$$\mathbf{P}_2 = \frac{1}{3}\mathbf{P}_0 + \frac{2}{3}\mathbf{P}_3 + h\|\mathbf{P}_0 - \mathbf{P}_3\|\mathbf{e}_v \tag{20}$$

where $\mathbf{e}_v$ is the unit vector in the upward vertical direction. Constant $h = 1/3$ means that the initial trajectory will have equal horizontal extent between knot points.

To optimize (16), we apply a secant version of the L–M algorithm [28], [35]. For the current trajectory generated by $\mathbf{x}$, we estimate the derivative $\nabla C(\mathbf{x})$ of the cost function $C(\mathbf{x})$ numerically, by sampling slightly modified trajectories $\mathbf{x}+\delta\mathbf{e}_j$, where $\mathbf{e}_j$, $1 \le j \le \dim(\mathbf{x})$, are the orthonormal bases, and we used $\delta = 10^{-1}$ in our implementation.

The secant version of the L–M algorithm iteratively builds a local quadratic approximation of $\{C(\mathbf{x})\}^2$ based on the numerical derivative, and then takes a step toward an improved state. The direction of the step is a combination of the steepest gradient descent direction and the conjugate gradient direction. We use the specific approach described by Madsen *et al.* [28, Sec. 3.5]. The iterative procedure terminates when the improvement in $\{C(\mathbf{x})\}^2$ becomes sufficiently small.

In the case of using multiple arms, we associate an individual trajectory $\mathbf{x}_i$ to each of the arms R_i. We then extend the state variable to $\mathbf{x} = (\mathbf{x}_1^T, \ldots)^T$. The rest of the optimization procedure is the same as the single arm case. Note that both single- and dual-arm trajectories are in the 3-D space. In Section VII-C, we describe the folding experiments using the method described in this section.

VII. Experiments

We used a series of experiments to demonstrate the effectiveness of the proposed method and justify the components. We tested our method on a data set of various kinds of garments collected from practical settings. Experimental results demonstrate that our method is able to achieve both reasonable accuracy and fast speed.

A. Pose Recognition

1) Data Acquisition: Since the simulated database in Section IV does not have the data captured in the real settings for domain adaptation (see Section V-C), we collect an extra

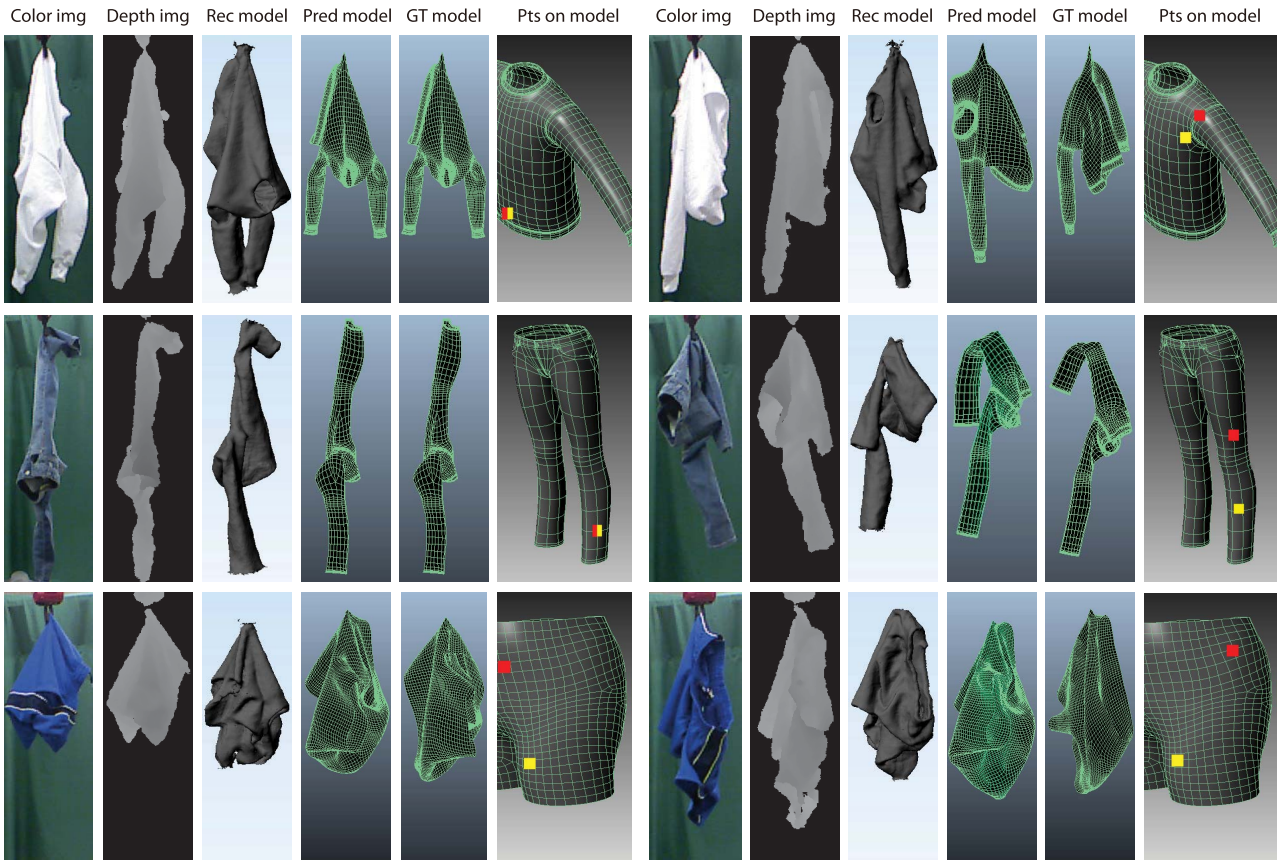


Fig. 12. Visual examples of the pose recognition result of our method. The garment is picked up via a griper of the Baxter robot. From left to right, each example shows the color image, input depth image, reconstructed model, matched simulated model, ground truth simulated model, and the predicted grasping points (red) marked on the model with the ground truth (yellow). Bottom-right: example is considered as a failure example, which may be because of the uninformative deformation shape. Note our method does not use any color information (best viewed in color).

test data set for general evaluation of pose recognition of deformable objects based on depth image as inputs.

The data set consists of two parts: a test set and a calibration set. To collect the testing set, we use a Baxter robot, which is equipped with two arms with seven degrees of freedom. A Kinect sensor is mounted on a horizontal platform at a height of 1.2 m to capture the depth images. We then use our 3-D reconstruction algorithm to reconstruct their mesh models. Given we also need to learn/calibrate a distance metric from extra data from Kinect [using (6)], we collect an extra small amount of data with the same settings as the calibration data, only collecting five poses for each garment. A weight vector $\boldsymbol{w}$ is then learned from this calibration data for each type of garment.

2) Qualitative Evaluation: We demonstrate some of the recognition results in Fig. 12 in the order of color image, depth image, reconstructed model, predicted model, ground truth model, and predicted grasping point (red) versus ground truth grasping point (yellow) on the garment. From Fig. 12, we can first see that our 3-D reconstruction is able to provide us with good-quality models for a fixed camera capturing a dynamic scene. And our shape retrieval scheme with learned distance metrics is also able to provide reasonable matches for the grasping points. Note that our method is able to output a mesh model of the target garment, which is critical to the subsequent operations, such as path planning and object manipulation.

Another observation is that the proposed recognition approach handles the noise well. It is not from the simulated database, because we did not add noise to the simulation of the mesh model, but more from the matching and recognizing stage, which is inherently robust. The robustness comes from: 1) the volumetric representation in both KinectFusion and binary feature extraction and 2) the weighted Hamming distance metric bridging the ideal simulated data and the real data.

3) Quantitative Evaluation:

a) Implementation details: In the 3-D reconstruction, we set $X = 384$ and $Y = Z = 768$ voxels and the resolution of the voxels as 384 voxels per meter to obtain a tradeoff between resolution and robustness against sensor noise. In the feature extraction, our implementation adopts $R = 16, \Phi = 16$, and $N = 16$ in the feature extraction as an empirically good configuration. That is, each mesh model gives a $16 \times 16 \times 16 = 4096$ dimensional binary feature. We set the penalty $C = 10$ in (6).

b) Classification accuracy: For each input garment, we compute the classification accuracy of pose recognition, that is

$$\text{Accuracy} = \frac{\#\text{ of correctly classified test cases}}{\#\text{ of all test cases}}. \quad (21)$$

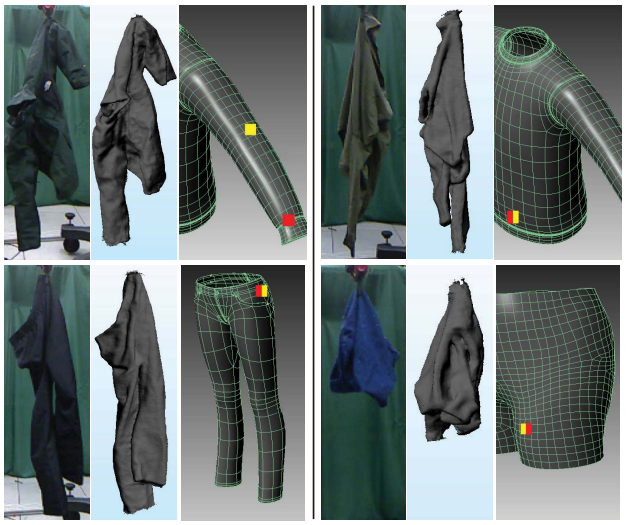


Fig. 13. Sample results of applying our method on novel garments. Each group of results shows the color image, reconstructed model, and predicted grasping points (red) versus ground truth (yellow) marked on the model from left to right (best viewed in color).

The classification accuracy for each garment type is reported in Table I (left). Given we have two models for each garment in the database (except shorts), we report the accuracy achieved using only Model 1 for retrieval, using only Model 2 for retrieval, and use all the available data. The total grasping points for sweaters, pants, and shorts are 19, 12, and 8, respectively. Our method is benefited from the 3-D reconstruction step, which reduces the sensor noise and integrates the information of each frame to a comprehensive model and thus leads to better decisions. Among three types of garments, recognition of shorts is not as accurate as the other two. One possible reason is that many of the shapes from different grasping points look very similar. Even for human observers, it is hard to distinguish them.

4) Generality to Novel Garments: Though we used a relatively small garment database for our experiments, we noticed that our simulated models can also be generalized to recognize similar but unseen garments. Fig. 13 shows some examples of recognizing poses of unseen garments using the same weight $\boldsymbol{w}$ learned on our original data set. We also noticed that there exist some decorations, such as pockets or shoulder boards on those garments; however, our method is robust enough to ignore these subtler features.

The reason why the proposed approach can still perform well on the unseen models is the visual similarity among models within the same category, and sometimes even the models across different categories. For example, sweaters and jackets can be considered as garments similar to our sweaters' model. Also, knit pants and suit pants are similar to our jeans model. Although they are made of different materials, the ways that they deform are similar to our training models for certain poses. This generality to unseen models may help reduce the burden of training data collection; however, it may not always be satisfactory from an end-to-end system perspective. For example, jackets may require a different way of manipulation from shirts. In such cases, more fine-grained categorization is preferred. Fortunately, in industrial applications, the categories are usually predetermined, and such undesirable conditions can be limited.

Fig. 14. Registration examples. First row: sweater grasped at elbow. Second row: long-sleeve shirt grasped at sleeve end. Third row: pair of pants grasped near knee. Fourth row: pair of pants grasped near ankle. Each row depicts from left to right: reconstructed mesh, predicted mesh from the database, rigid registration only, and rigid plus nonrigid registration.

B. Registration and Iterative Regrasping

1) Registration: We apply both rigid and nonrigid registrations, while rigid step focuses on mesh rescaling and alignment, and the nonrigid step refines the mapping accuracy. In Fig. 14, we compare the difference between using rigid registration only and using rigid plus nonrigid registration side by side. We can clearly see that with nonrigid registration, the two meshes are registered more accurately. In addition, the location of the designated grasping points on the sleeves is also closer to the ground truth points. Note that for the fourth

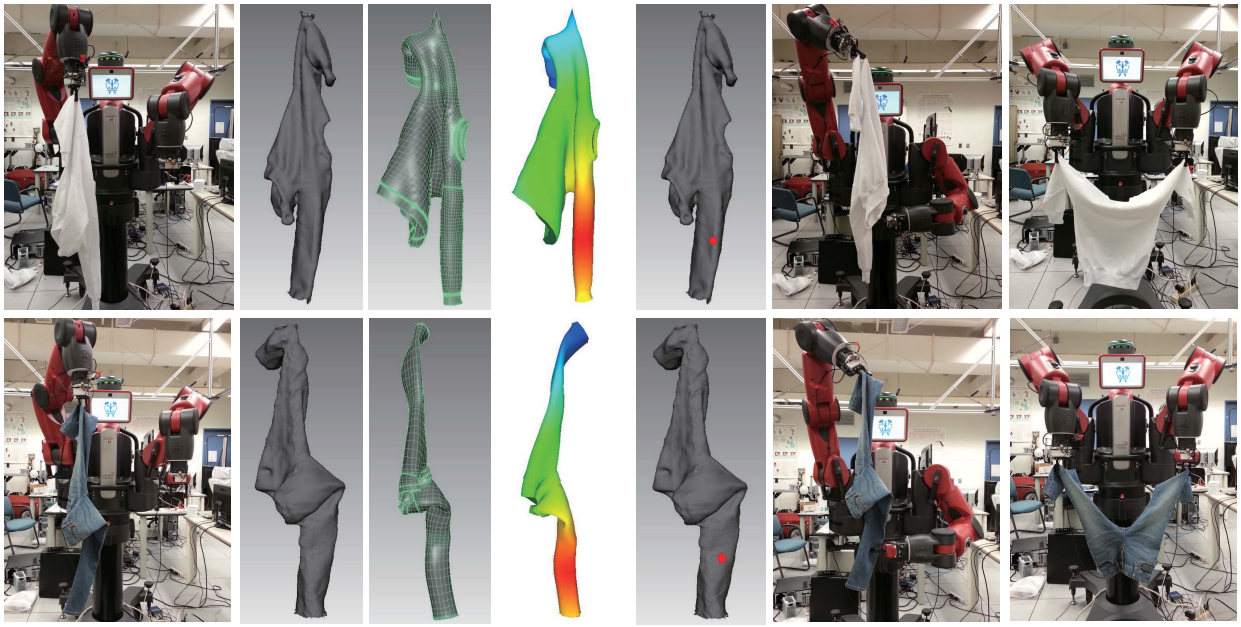


Fig. 15. Examples of each step in our regrasping procedure. For each row from left to right: snapshot of initial pickup, predicted mesh from database, predicted mesh with weighted Gaussian distribution distance, predicted regrasping point on the 3-D reconstructed mesh, and finally snapshot of unfolding. Top row: Baxter robot unfolds a sweater following pickup. Bottom row: Baxter robot unfolds a pair of pants following pickup.

TABLE I

LEFT: AVERAGE CLASSIFICATION ACCURACY FOR DIFFERENT GARMENT TYPES. RIGHT: AVERAGE RUNNING TIME (EXCLUDING 3-D RECONSTRUCTION) IN SECONDS TO PROCESS ONE GARMENT OF THE PROPOSED METHOD ON THE PROPOSED DATABASE, WITH THE INPUT OF DIFFERENT GARMENT TYPES

Garment	Model 1	Model 2	Both models
sweaters	63.1%	68.4%	73.7%
Pants	75.0%	75.0%	75.0%
Shorts	62.5%	N/A	62.5%

Garment	Running Time
sweaters	0.30
Pants	0.20
Shorts	0.22

row, after the alignment by the rigid registration algorithm, the state is evaluated as a local minimum. Therefore, there is no improvement by the following nonrigid registration. But as we can see from the visualization, such a case is still good enough for finding point correspondence.

We also evaluate the registration algorithm on the entire database, which contains two stages: rigid registration using ICP algorithm and nonrigid registration algorithm. To show the performance of our registration algorithm, the registration pairs are established with the knowledge that the recognition of the pose is 100% correct. This will enable the registration to happen between the closest grasping locations. Meanwhile, we design the registration experiments in two directions: the source mesh to the target mesh, and vice versa. We also compare the registration results of the rigid registration and the rigid plus the nonrigid registration for all the pairs. Detailed results are shown in Table II. For example, for the S to T(R), we first subdivide the source mesh into a set of disjoint triangulated patches, and generate a single sample point in each patch. Each sample point is also assigned the area of the patch it belongs to. Then, from each such sample point, we find the closest point on the target mesh, and sum up the distance of all point pairs and multiplied by the corresponding patch area. Finally, the summed value is divided by the total area of the source mesh.

TABLE II

REGISTRATION RESULTS. WE COMPARE THE SOURCE MESH (S) REGISTERED TO THE TARGET MESH (T), AND VICE VERSA, FOR BOTH RIGID-ONLY REGISTRATION (R) AND RIGID PLUS NONRIGID REGISTRATION (R + N). WE CAN SEE THAT WHEN THE SOURCE MESH REGISTERED TO THE TARGET MESH, THE AVERAGE ERROR DISTANCE IS LESS THAN THE TARGET MESH REGISTERED TO THE SOURCE MESH. ALSO, WE CAN SEE THAT WITH ADDITIONAL NONRIGID REGISTRATION, THE AVERAGE ERROR DISTANCE IS REDUCED

Source Mesh	S to T (R)	T to S (R)	S to T (R+N)	T to S (R+N)
Sweater 1	0.0251	0.0188	0.0073	0.0129
Sweater 2	0.0171	0.0218	0.0151	0.0213
Pants 1	0.0099	0.0652	0.0060	0.0648
Pants 2	0.0164	0.0221	0.0044	0.0172
Shorts	0.0238	0.0149	0.0081	0.0092
Average	**0.0185**	**0.0256**	**0.0070**	**0.0250**

2) Iterative Regrasping: Fig. 15 shows two examples (sweater and pants) of iterative regrasping using the Baxter robot. The robot first picks up a garment at a random grasping point. Once the arm reaches a predefined position, the last joint of the arm starts to rotate and the Kinect will capture the depth images as it rotates, and reconstruct the 3-D mesh in real

Garment	# of Trial	Successful Recognition	Successful Regrasping	Successful Unfolding	Avg. # of Regrasps Success Only
Sweatshirt	10	9/10	8/10	8/10	1.6
Sweater	10	8/10	8/10	7/10	1.6
Knitwear	10	9/10	9/10	8/10	1.7
Jeans	10	9/10	9/10	9/10	1.3
Pants	10	8/10	10/10	9/10	1.4
Leggings	10	8/10	9/10	8/10	1.4
Shorts	10	7/10	8/10	7/10	1.9
Average	10	8.3/10	8.7/10	8.0/10	1.6

Fig. 16. Left: picture of our test garments. Right: results for each unfolding test on the garments. We evaluate the results by recognition, regrasping, unfolding, and regrasping attempts for each test. The last row shows the average of each evaluation component.

time. After the rotation, a predicted pose is recognized [24] as shown in the third image of each row. For each pose, we have a constrained weighted evaluation metric over the surface to identify the regrasping point as indicated in the fourth image. By registration of the reconstructed mesh and predicted mesh from the database, we can map the desired regrasping point onto the reconstructed mesh. The robot then regrasps by moving the other gripper toward it. With our 1-D blob curvature detection method, the gripper can move to the best curvature on the garment and regrasp, which increases the success rate. The iterative regrasping stops when the two grasped points are the designated anchor points on the garment (e.g., elbows on the sleeves of a sweater).

Fig. 16 (left) shows seven sample garments in our test, and the table on the right shows the results. For each garment, we perform ten unfolding tests. We have on average an 83% successful recognition rate for the pose of the objects over all the garments. We have on average an 87% successful regrasping rate for each garment, where regrasping is defined as a successful grasp of the other arm on the garment. We are able to achieve a successful rate of 80% to unfold the garment, placing the grippers at the designated grasping points. Unsuccessful unfolding occurred when either the gripper lost contact with the garment, or the gripper was unable to find a regrasping point. Although we did not perform this experiment, it is possible to restart the method after one of the grippers loses contact as an error recovery procedure.

For the successful unfolding cases, we also report the average number of regrasping attempts. The minimum number of regrasping attempts $= 1$. This happens when the initial grasping is at one of the desired positions, and the regrasping succeeds at the other desired position (i.e., two elbows on the sleeves for a sweater). In most cases, we are able to successfully unfold the garments using 1–2 regraspings.

Among all these garments, jeans, pants, and leggings achieve a high success rate because of their unique layout when grasping at the leg position. The shorts are difficult for both recognition and unfolding steps possibly because of their ambiguous appearances in different grasping points. One observation is that in a few cases, when the recognition is not accurate, our registration algorithm was sometimes able to find a desired regrasping point for unfolding. This is an artifact of the geometry of pantlike garments where the designated regrasping points are at the extreme locations on the garments. It is important to note that the gripper of the Baxter robot has a limit of grasping capabilities. For garments such as thick jackets and jeans, droppings may happen in the process. Therefore, we select relatively light garments in our experiments. We also attached rubbers onto gripper to increase the friction.

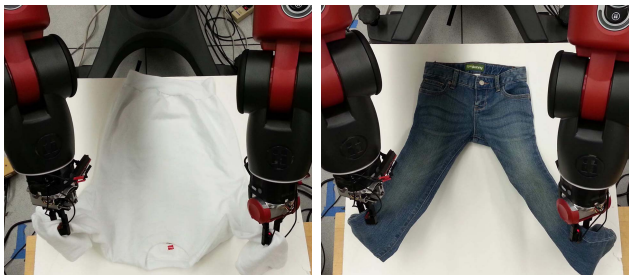


Fig. 17. Baxter robot places a garment flat on a table. Left: garment is a sweater and the two desired grasping points are on the sleeves. Right: garment is a pair of pants and the two desired grasping points are on the lower leg parts.

3) Laying Flat on a Table: We also show that after grasping at two desired points, the robot will proceed to place the garment on a table. In our experiments, we use cardboard to simulate a table area. As shown in Fig. 17, the robot is able to place the garment flat with a simple move when grasping at a pair of two desired grasping points. With such a flat configuration, the robot can begin to fold it. Practically, in terms of the garment size, we notice that sometimes it is difficult to achieve a fully flattened garment on the table by the maximum opening of the two arms of the Baxter robot. Therefore, in such cases, manual adjustments may be needed to arrive at a good standing point of the coming folding step.

C. Folding by Minimizing Distance

1) Parameter Adaptation: Two key parameters affect the how realistic the simulation is, the material properties of the fabric, and the frictional forces between the garment and the table.

a) Material properties: Through many experiments, we found that the most important property for the garments in the simulation environment is shear resistance. It specifies the amount that the simulated mesh model resists shear under strain; when the garment is picked up and hung by gravity,

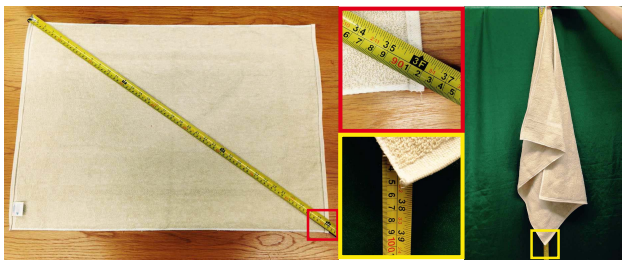


Fig. 18. Method for measuring the shear resistance. Left: diagonal length measurement. Middle: zoomed-in regions. Right: garment is hanging under gravity.

the total length will be elongated due to the balance between gravity force and shear resistance. An appropriate shear resistance measure allows the simulated mesh to reproduce the same elongation as the real garment. This measurement will bridge the gap between the simulation and the real world for the garment mesh model.

For each new garment, we follow the following steps to measure the shear resistance. Fig. 18 shows an example.

- Manually pick one extremum part of the garment, such as the sleeve end of a sweater, the waist part of a pair of pants, and a corner of a towel.
- Hang the garment under gravity and measure the length between the picking point and the lowest point as L_1.
- Slowly put down the garment on a table and keep the picking point and the lowest point in the previous step at maximum spread condition. Measure the distance between these two points again as L_2. The shear resistance fraction is defined by the following:

$$\text{shear\_frac} = (L_1 - L_2)/L_2. \tag{22}$$

- We then pick up and hang the virtual garment into the same configuration in *Maya*, adjusting the *Maya* shear parameter, such that the shear fraction as calculated in the simulator is identical to the real world.

b) Frictional forces: The surface of the table can be rough if covered by a cloth sheet or slippery if not covered, which leads to variance in friction between the table and the garment. A shift of the garment during the folding can possibly impair the whole process and cause additional repositioning. Adjusting the frictional level in the simulation environment to the real world is crucial and necessary for trajectory optimization.

To measure the friction between the table and the garment, we do the following steps.

- Place a real garment on the real table of length L_t.
- Slowly lift up one side of the real table, until the garment in the real world begins to slide. The lifted height is H_s. The friction angle is computed as

$$\angle_{\text{Friction}} = \sin^{-1}(H_s/L_t) \tag{23}$$

- In the virtual environment, the garment is placed flat on a table with gravity. Assign a relatively high friction value to the virtual table. Lift up one side of the virtual table to the angle of $\angle_{\text{Friction}}$.
- Gradually decrease the frictional force in the virtual environment, until the garment begins to slide. Use this frictional force in the virtual environment as it mirrors the real world.

With these two parameter setups, we obtain similar manipulation results for both the simulation and the real garment. Fig. 19 (left) shows a picture of all the test garments we used in different colors, sizes, and materials. The table on the right of Fig. 19 shows the measured parameters of each test garment, including stretch percentage and friction angle, and corresponding *Maya* parameters. For common garments, these parameters do not have a significant variance. Therefore, we suggest that if researchers use simulators such as *Maya*, the average values are a reasonably good start.

2) Garment Manipulation and Folding: Fig. 20 shows three successful folding examples from the simulation and the real world, including a sweater, a pair of pants, and a medium size towel. We show six key frames for each folding task. The folding poses from the simulation are in the first row of each group with an optimized trajectory. We also show the corresponding results from the real world. The green tape on the table indicates the original contour of the garment.

Each garment is first segmented from the background, and key points are detected from the binary mask. Given the key points, a corresponding multistep folding plan is created (the folding plan is predefined, and one of our folding plans for a sweater is shown in Fig. 21). For each garment, we have optimized trajectories for each folding step. Here, we map these optimized trajectories to our scenario according to the generated folding plan. Then, the Baxter robot follows the folding plan with optimized trajectories to fold the garment. We can see that the deformation of the real garment and the simulated garment is very similar. Therefore, the final folding outcome is comparable with the simulation.

Table III shows the statistical results of the garment folding test. Each time one or two robotic arms fold the garment counts as one fold. We ran ten trials for each test garment. It turns out that the folding performance of the sweaters and towels are very stable with our optimized trajectories. Jeans and pants are less stable, because the shear resistance of the surface is relatively high, and sometimes is difficult to bend, leading to unsuccessful folding. In the successful folding cases for jeans and pants, we sometimes ended up with small wrinkles, but the folding plan was still able to complete successfully.

We also show the average time to fold a garment in the last row. The robot is able to fold most garments in about 1.5 min. Meanwhile, we found that it is challenging, because the garments have to be well placing-flat on the table within a range. Any misalignment may lead to a different or even a wrong folding plan. For safety purpose, the robot arms move relatively slow in the experiments. The folding action could be potentially sped up in a standard industrial-level pipeline.

3) Solution Space: The solution space is a subspace of the trajectory space where the folded garment ends in a shape with a dissimilarity score less than a threshold. Intuitively, a number of trajectories within the solution space will fold

Garment Type	Stretch (%)	Friction Angle (°)	Maya Shear Resistance	Maya Friction
Sweater (large)	2.9	24.3	200	0.7
Sweater (small)	2.9	24.7	200	0.7
Jeans	2.9	19.1	200	0.5
Pants	1.7	21.9	340	0.6
Large Towel	2.2	18.7	260	0.5
Medium Towel	3.1	22.3	190	0.6
Small Towel	1.1	24.3	530	0.7
Average	**2.4**	**22.2**	**274**	**0.6**

Fig. 19. Left: picture of our test garments. Right: results for each unfolding test on the garments. We show the results of stretch percentage, friction angle of the table, and the corresponding parameters in Maya by each test. The last row shows the average of each measurement component.

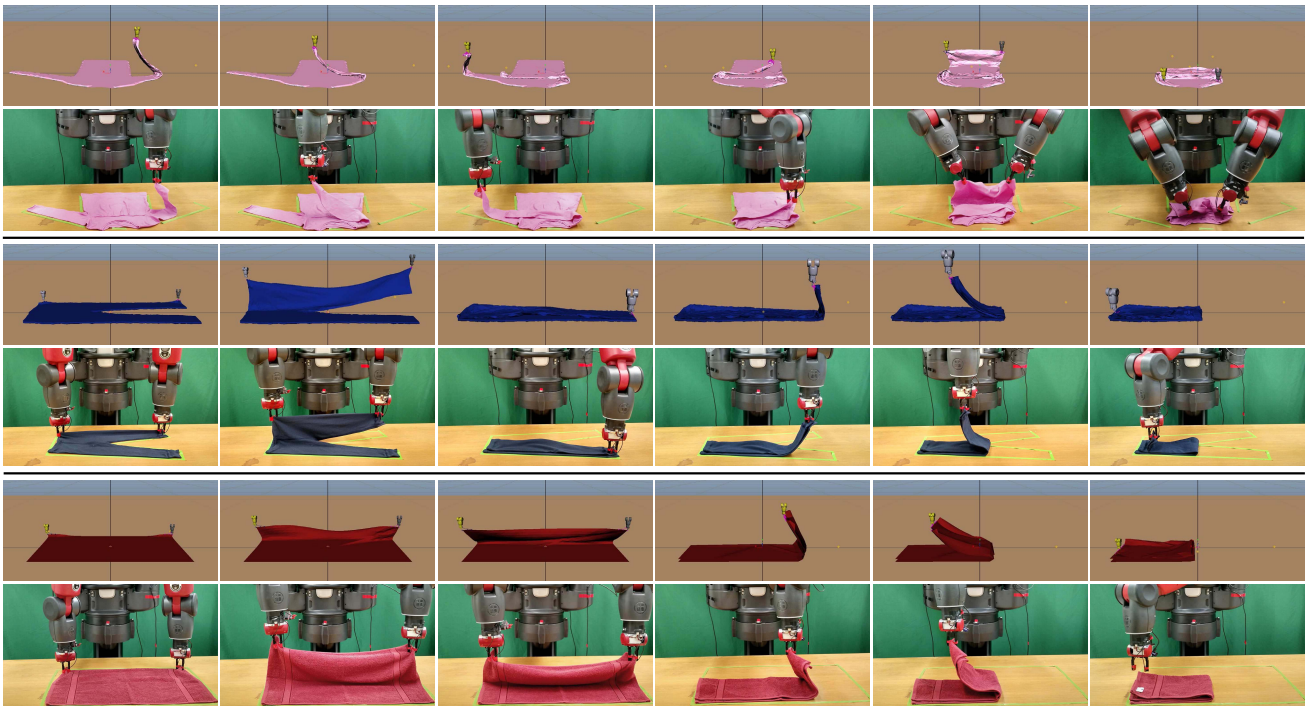


Fig. 20. Successful folding examples with optimized folding trajectories from off-line simulation. The first row of each group is from the simulation and the second row is from the real world (green tape shows the original garment contour position). Top: sweater folding with three steps. Middle: pants folding with two steps. Bottom: medium size towel folding with two steps.

the garment, leaving its shape close to the desired shape. We have found that trajectories within the solution space can vary to a degree while still allowing the robot to accomplish the folding task. This result also agrees with the fact that people do not have to follow a unique trajectory to fold the garment. However, trajectories outside the solution space cause issues for the folding tasks. Our trajectory optimization automatically avoids such cases.

To further explore the relationship between the trajectories and folded shapes, we experimented the folding with a few different trajectories in simulation. A notable finding is that the symmetric trajectories can always produce better folded shape, as shown in Fig. 22. The 13 color curves in each plot represent 13 different trajectories. The dissimilarity bar on the right shows the difference between the folded shape and the desired folded shape for each folding simulation. We also tested with asymmetric trajectories for the folding, as shown in the second and third plots in Fig. 22. We can see that the

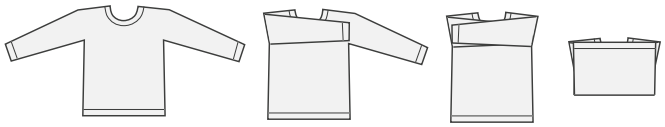


Fig. 21. Garment folding plan for a sweater.

second plot has larger dissimilarities than the first and third, which is mainly caused by the friction. The robot should raise the starting point to a high enough position at the beginning to prevent the grasped portion of the garment pushing the other portion on the table. This is also consistent with our simulation results that our optimizer will drive the height of the trajectories to a reasonable distance from the garment.

There is a tradeoff between doing contour fitting at each step and total time spent to fold a garment. In this paper, we start with one template and then assume that each step after

TABLE III

RESULTS OF FOLDING TEST FOR EACH GARMENT. WE SHOW THE NUMBER OF FOLDING STEPS, SUCCESSFUL RATE, AND TOTAL TIME OF EACH GARMENT. EACH GARMENT HAS BEEN TESTED TEN TIMES. THE TIME IS THE AVERAGE OVER ALL SUCCESSFUL TRIALS FOR EACH GARMENT

Garment Type	# of folds	Success Rate	Avg. Time (sec)
Sweater (large)	3	10/10	121
Sweater (small)	3	10/10	118
Jeans	2	7/10	88
Pants	2	8/10	88
Large Towel	2	10/10	90
Medium Towel	2	10/10	88
Small Towel	2	10/10	83
Average	**2.3**	**9.3/10**	**97**

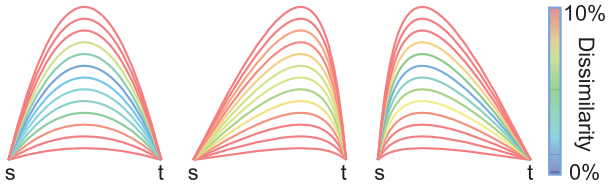


Fig. 22. Dissimilarity values from different trajectories for folding the towel model in the second folding step. The trajectory is projected to a 2-D plane for illustration purposes. S and T stand for the start and the target position, respectively (best viewed in color).

that the folded garment is close to that in the simulation. Our experimental results, as shown in Table III, verify that this method works well and is able to save time, since we only do the contour fitting once. With our simulated trajectories, the Baxter robot is able to fold a garment under predefined steps correctly. An alternative method could use the contour fitting at each step, but this would require more time and computation.

We note that some failures due to the motor control error from the Baxter robot. When the robot executes an optimized trajectory, its arm suffers from a sudden drop or jitter. Such actions will raise pull forces to the garment, leading to drift and inaccurate folding. This can be solved by using an industrial-level robotic arm with more accurate control. We also note that failures can be recognized with the correct sensing suite, and we are currently investigating ways to effect online error recovery for such failures. One difference between the simulation and the real world we found is that moving a point on the mesh in the simulation is different from using a gripper to grasp a small area of a real garment and move it. In the future, we hope to be able to simulate a similar grasp effect for the trajectory optimization.

VIII. CONCLUSION

In this paper, we introduced a simulation database of common deformable garments to facilitate recognition and manipulation. The database contains five different garments within three categories: sweater, pants, and shorts. Each garment is fully simulated with a number of depth images and 3-D mesh models for all the semantic labeled grasping points. We demonstrated a consistent and complete pipeline of manipulating a deformable garment to a target pose. First, from a mathematical optimization perspective, we derive the high-level framework as a two-stage algorithm, recognition, and manipulation. In recognition, we treat the pose estimation as a 3-D shape retrieval problem, and transfer the known pose from the most similar model in the database to the observed model. Then, rigid and nonrigid registration is performed to support the manipulation step. In manipulation, we import the mesh model into the simulator and compute the optimized trajectories. We extensively tested each component of the pipeline with designed experiments, such as garment recognition via picking up, unfolding the garment to a known desired state and laying flat, and using precomputed folding plans to fold it using a novel trajectory optimization method that prevents common folding errors. We have addressed all the phases of the pipeline in Fig. 1 individually. However, there are still some system and hardware issues that prevent the system from being a completely seamless pipeline. This is mainly due to kinematic constraints on the Baxter robot, which limits its ability to work with larger garments on a normal size table.

While the focus of this paper has been on clothing, we want to underline the point that model-driven, feedforward prediction can work well in complex environments with many unknown states. We believe that the ideas in this paper can be ported to similar domains, such as food handling ("soft deformable objects") and articulated rigid objects with multiple kinematic states. Furthermore, the step of placing the garment flat on the table is less explored. Currently, we use action playback for the step, but more factors need to be considered if more flexibility is desired.

ACKNOWLEDGMENT

The authors would like to thank J. Weisz, J. Varley, and R. Ying for many discussions, and P. M. Lopez for the help of the folding plan. They would also like to thank Nvidia Corporation and Intel Corporation for the hardware support.

REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 24, pp. 509–522, Apr. 2002.

[2] P. J. Besl and D. N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[3] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 113-1–113-16, Jul. 2013.

[4] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Proc. ICRA*, 2011, pp. 3893–3900.

[5] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *Proc. ICRA*, May 2014, pp. 987–993.

[6] G. E. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*. San Diego, CA, USA: Academic, 1988.

[7] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proc. ECCV*, 2004, pp. 224–237.

[8] A. Gabas, E. Corona, G. Alenyà, and C. Torras, "Robot-aided cloth classification using depth information and CNNs," in *Proc. Int. Conf. Articulated Motion Deformable Objects*, 2016, pp. 16–23.

[9] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Proc. ACM SIGGRAPH/Eurogr. Symp. Comput. Animation (SCA)*, Aire-la-Ville, Switzerland, 2003, pp. 62–67.

[10] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, Sep. 1993.

[11] AutoDesk Inc. *Maya 2015*. Accessed: 2013. [Online]. Available: https://en.wikipedia.org/wiki/Autodesk_Maya

[12] Poser World Inc. (2016). *Poser World Clothes Models*. [Online]. Available: http://www.poserworld.com/

[13] Turbo Squid Inc. (2016). *Turbo Squid 3D Models*. [Online]. Available: http://www.turbosquid.com/

[14] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. KDD*, 2002, pp. 133–142.

[15] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Clothes handling based on recognition by strategic observation," in *Proc. Humanoid Robots*, 2011, pp. 53–58.

[16] Y. Kita and N. Kita, "A model-driven method of estimating the state of clothes for manipulating it," in *Proc. WACV*, 2002, pp. 63–69.

[17] Y. Kita, T. Ueshiba, E. S. Neo, and N. Kita, "Clothes state recognition using 3D observed data," in *Proc. ICRA*, 2011, pp. 1220–1225.

[18] L. J. Latecki, R. Lakamper, and T. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Proc. CVPR*, 2000, pp. 424–429.

[19] T.-H.-L. Le, M. Jilich, A. Landini, M. Zoppi, D. Zlatanov, and R. Molfino, "On the development of a specialized flexible gripper for garment handling," *J. Autom. Control Eng.*, vol. 1, no. 3, pp. 255–259, 2013.

[20] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," in *Proc. Symp. Geometry Process. (SGP)*, Aire-la-Ville, Switzerland, 2008, pp. 1421–1430.

[21] H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev, "3D self-portraits," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013, Art. no. 187.

[22] Y. Li, C.-F. Chen, and P. K. Allen, "Recognition of deformable object category and pose," in *Proc. ICRA*, May/Jun. 2014, pp. 5558–5564.

[23] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen, "Multi-sensor surface analysis for robotic ironing," in *Proc. ICRA*, 2016, pp. 5670–5676.

[24] Y. Li, Y. Wang, M. Case, S.-F. Chang, and P. K. Allen, "Real-time pose estimation of deformable objects using a volumetric approach," in *Proc. IROS*, Sep. 2014, pp. 1046–1052.

[25] Y. Li *et al.*, "Regrasping and unfolding of garments using predictive thin shell modeling," in *Proc. ICRA*, May 2015, pp. 1382–1388.

[26] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *Proc. IROS*, 2015, pp. 6000–6006.

[27] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. ICCV*, 1999, pp. 1150–1157.

[28] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems* (Informatics and Mathematical Modelling), 2nd ed. Lyngby, Denmark: Technical Univ. Denmark, 2004, p. 60.

[29] 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 *Proc. ICRA*, 2010, pp. 2308–2315.

[30] I. Mariolis, G. Peleka, A. Kargakos, and S. Malassiotis, "Pose and category recognition of highly deformable objects using deep learning," in *Proc. Int. Conf. Adv. Robot. (ICAR)*, Jul. 2015, pp. 655–662.

[31] (2014). *Maya*. [Online]. Available: http://www.autodesk.com/products/autodesk-maya/

[32] S. Miller, J. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 249–267, 2012.

[33] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *Proc. ICRA*, Sep. 2011, pp. 4861–4868.

[34] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. ISMAR*, 2011, pp. 127–136.

[35] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. Berlin, Germany: Springer, 2006.

[36] R. Osada and T. Funkhouser, "Matching 3D models with shape distributions," in *Proc. SMI Int. Conf.*, 2001, pp. 154–166.

[37] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of massive laundry and classification types by dual manipulator," *J. Adv. Intell. Informat.*, vol. 11, no. 5, pp. 457–463, 2007.

[38] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *Proc. ICRA*, May 2013, pp. 1130–1137.

[39] J. Stria, D. Průša, and V. Hlaváč, "Polygonal models for clothing," in *Proc. Towards Auto. Robot. Syst.*, 2014, pp. 173–184.

[40] J. Stria *et al.*, "Garment perception and its folding using a dual-arm robot," in *Proc. IROS*, Sep. 2014, pp. 61–67.

[41] L. Sun, G. Aragon-Camarasa, S. Rogers, and P. Siebert, "Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening," in *Proc. ICRA*, 2015, pp. 185–192.

[42] G. K. L. Tam *et al.*, "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 7, pp. 1199–1217, Jul. 2013.

[43] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *Proc. CVPR*, Jun. 2003, pp. I-127–I-133.

[44] A. Toshev, B. Taskar, and K. Daniilidis, "Object detection via boundary structure segmentation," in *Proc. CVPR*, 2010, pp. 950–957.

[45] Y. R. Tsai, "Rapid and accurate computation of the distance function using grids," *J. Comput. Phys.*, vol. 178, no. 1, pp. 175–195, 2002.

[46] Z. Tu and A. L. Yuille, "Shape matching and recognition: Using generative models and informative features," in *Proc. ECCV*, 2004, pp. 195–209.

[47] N. Umetani, D. M. Kaufman, T. Igarashi, and E. Grinspun, "Sensitive couture for interactive garment editing and modeling," *ACM Trans. Graph.*, vol. 30, no. 4, Aug. 2011.

[48] J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel, "Gravity-based robotic cloth folding," in *Proc. Int. Workshop Algorithmic Found. Robot. (WAFR)*, 2010, pp. 409–424.

[49] J. Wang, L. Yin, X. Wei, and Y. Sun, "3D facial expression recognition based on primitive surface feature distribution," in *Proc. CVPR*, 2006, pp. 1399–1406.

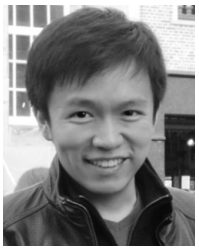
[50] P.-C. Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbbeel, "Perception for the manipulation of socks," in *Proc. IROS*, 2011, pp. 4877–4884.

[51] Y. Wang, R. Ji, and S.-F. Chang, "Label propagation from imagenet to 3D point clouds," in *Proc. CVPR*, Jun. 2013, pp. 3135–3142.

[52] B. Willimon, S. Birchfield, and I. Walker, "Classification of clothing using interactive perception," in *Proc. ICRA*, 2011, pp. 1862–1868.

[53] B. Willimon, I. Walker, and S. Birchfield, "A new approach to clothing classification using mid-level layers," in *Proc. ICRA*, 2013, pp. 4271–4278.

[54] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys, "3D model matching with viewpoint-invariant patches," in *Proc. CVPR*, 2008, pp. 1–8.

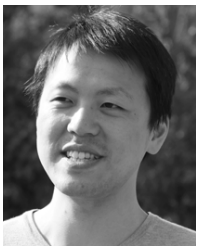


Yinxiao Li received the Ph.D. degree in computer vision, machine learning, with application to robots from Columbia University, New York, NY, USA, in 2016.

He then joined Google as a Software Engineer. His Ph.D. research project on laundry robot has been featured on media such as the DailyMail, NVidia research blog, and *Communications of the ACM*, and selected in the conference trailer at ICRA.

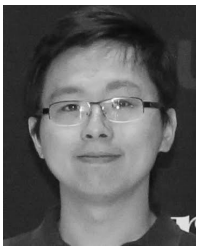
Yan Wang received the Ph.D. degree in 3D/2D content understanding and search from Columbia University, New York, NY, USA, in 2015.

He then joined Microsoft to develop new image understanding techniques in Bing, especially on fast object detection and instance recognition, both on web scale images. His algorithms have been patented and integrated in Bing Image Search, Facebook Graph Search and Adobe Photoshop, and featured on media such as NBCNews and MIT Technology Review.



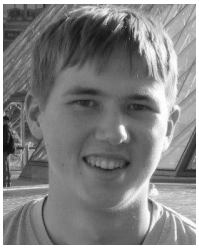
Yonghao Yue received the B.S. and M.S. degrees in 2005 and 2007, and Ph.D. degree in information science and technology from The University of Tokyo, Tokyo, Japan, in 2011.

He was a Research Scientist with Columbia University, New York, NY, USA, until 2016. He is currently an Assistant Professor with the Department of Complexity Science and Engineering, Graduate School of Frontier Sciences, The University of Tokyo. His current research interests include physically based rendering and simulation and their applications for designing functional man-made objects and robotics.



Danfei Xu received the B.S. degree in computer science from Columbia University, New York, NY, USA, in 2015. He is currently pursuing the Ph.D. degree with the Computer Science Department, Stanford University, Stanford, CA, USA.

His research interests include robotic learning and 3-D computer vision.



Michael Case received the B.S. degree from Columbia University, New York, USA, in 2014.

He interned at Microsoft Redmond, Redmond, WA, USA. He is currently a Software Engineer at Google.

Shih-Fu Chang (F'04) is the Senior Executive Vice Dean and the Richard Dicker Professor of Columbia Engineering. His research interests include multimedia information retrieval, computer vision, machine learning, and signal processing, with the goal to turn unstructured multimedia data into searchable information.

Prof. Chang has been awarded the IEEE Signal Processing Society Technical Achievement Award, ACM Multimedia SIG Technical Achievement Award, Honorary Doctorate from the University of Amsterdam, and IEEE Kiyo Tomiyasu Award. He is a fellow of the American Association for the Advancement of Science and ACM.

Eitan Grinspun received the B.S. degree in engineering science from the University of Toronto, Toronto, ON, Canada, in 1997, and the M.S. and Ph.D. degrees in computer science from the California Institute of Technology, Pasadena, CA, USA, 2000 and 2003, respectively.

He is an Associate Professor of computer science and applied mathematics at Columbia University, New York, NY, USA, and the Co-Director of the Columbia Computer Graphics Group. He was an Alfred P. Sloan Research Fellow and NSF CAREER Award recipient, NVIDIA Fellow, and a Caltech Everhart Distinguished Lecturer. Prior to joining Columbia University, he was a Research Scientist at the Courant Institute of Mathematical Sciences, New York, NY, USA, from 2003 to 2004, a doctoral student with the California Institute of Technology until 2003, and an undergraduate at the University of Toronto. He was profiled in The New York Times, Popular Science ("Brilliant 10 Scientists of 2011"), Fast Company ("Most Creative People in Business 2013"), Scientific American, New Scientist, and mentioned in Variety. The NSF-funded technologies developed by his laboratory are found in Adobe Photoshop and Illustrator, major film studios including Disney, Pixar, and Weta Digital, and condensed matter physics laboratories. His film credits include The Hobbit, Rise of the Planet of the Apes, and Steven Spielberg's The Adventures of Tintin.

Peter K. Allen received the A.B. degree in mathematics-economics from Brown University, Providence, RI, USA, in 1970, the M.S. degree in computer science from the University of Oregon, Eugene, OR, USA, in 1976, and the Ph.D. degree in computer science from the University of Pennsylvania, Philadelphia, PA, USA, in 1985.

He is a Professor of computer science with Columbia University, New York, NY, USA, and the Director of the Columbia Robotics Lab. He was the recipient of the CBS Foundation Fellowship, Army Research Office Fellowship, and the Rubinoff Award for innovative uses of computers from the University of Pennsylvania. His current research interests include robotic grasping, 3-D vision and modeling, and medical robotics.