2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Recognition of Deformable Object Category and Pose

Yinxiao Li, Chih-Fan Chen, and Peter K. Allen

***Abstract*— We present a novel method for classifying and estimating the categories and poses of deformable objects, such as clothing, from a set of depth images. The framework presented here represents the recognition part of the entire pipeline of dexterous manipulation of deformable objects, which contains grasping, recognition, regrasping, placing flat, and folding. We first create an off-line simulation of the deformable objects and capture depth images from different view points as training data. Then by extracting features and applying sparse coding and dictionary learning, we build up a codebook for a set of different poses of a particular deformable object category. The whole framework contains two layers which yield a robust system that first classifies deformable objects on category level and then estimates the current pose from a group of predefined poses of a single deformable object. The system is tested on a variety of similar deformable objects and achieves a high output accuracy. By knowing the current pose of the garment, we can continue with further tasks such as regrasping and folding.**

I. Introduction

Deformable objects such as clothes, fabrics, paper, and things that are soft, are ubiquitous in our daily life. In order to enable robots to manipulate deformable objects, we first have to let the robots recognize them through perception. Compared with rigid object recognition which has finite state spaces, deformable object recognition is much harder because of the very large dimensional spaces. Each space may have completely different appearance in terms of the material and pose of the object. For example, in robot grasping, it is possible to identify the grasping point location on a rigid object by a shape matching algorithm from a known object in a training set. However, due to the large number of states of a deformable object, it is much more difficult to identify the pose and grasping point.

In this paper, we are focusing on the recognition of deformable object category and pose as one component of a larger pipeline for manipulating deformable objects such as clothing. The main idea of our method is to use off-line simulation of models of garments in different poses to predict online category and poses. Key contributions are:

- A method for estimating deformable object pose using simulation results and dictionary learning via spatial pyramid matching and sparse coding.
- A system that captures 90 depth images from different view points of a target object in $20-50$ different poses. Figure 1 shows two garment mesh models and one of their poses hanging by gravity.
- Formulation of the pose recognition problem as a hierarchical image classification task that uses depth images regardless of complicated texture on the object.
- Experimental results in simulation, with real clothing, and also with a physical robot on a variety of deformable objects including sweaters, jeans, and shorts in different sizes. The results show that our approach is robust to estimate the object pose and to facilitate regrasping and folding tasks.

Y. Li, C. Chen, and P. K. Allen are with the Department of Computer Science, Columbia University, New York, NY 10027, USA {yli@cs., cc3500@, allen@cs. } columbia.edu

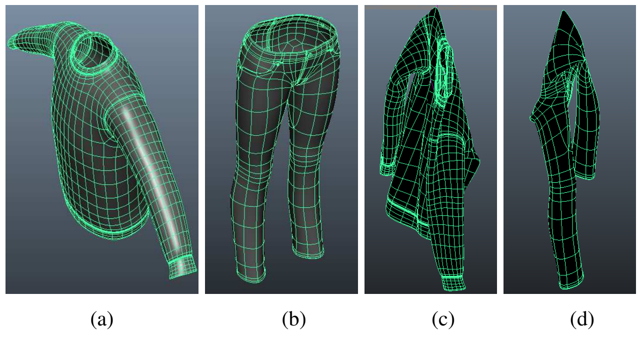
Fig. 1. (a), (b): Garment mesh models of sweater and jeans rendered in *Maya*. (c), (d): Simulation of hanging under gravity of the garment mesh models, respectively.

II. Related Work

A significant amount of previous research has focused on deformable object detection. In these works, the primary goal is to detect the deformable objects in the scene via some features, but they do not explicitly compute object pose. Felzenszwalb, et. al [3] described an algorithm based on mixtures of multi-scale deformable part models trained using a discriminative procedure. Another fast deformable object detection approach was proposed by Pedersoli, et. al [16], which improved the part-based model by doing a multiple-resolution hierarchical layer structure which increased the detection accuracy. Though their work has very promising detection accuracy on many different object categories, it still does not address the pose recognition problem.

There has been previous work on the recognition and manipulation of deformable objects. B. Willimon, et. al [22], [21] used interactive perception to classify the clothing type. However, their work basically focused on small clothing such as socks and shorts which usually consist of a single color. The proposed method may fail with some fully textured clothes because it relies on a color-based image segmentation algorithm. Miller, et. al [15], Wang, et. al [20], Schulman, et. al [17], Cusumano-Towner, et. al [1] have done some impressive work in clothing recognition and manipulation.

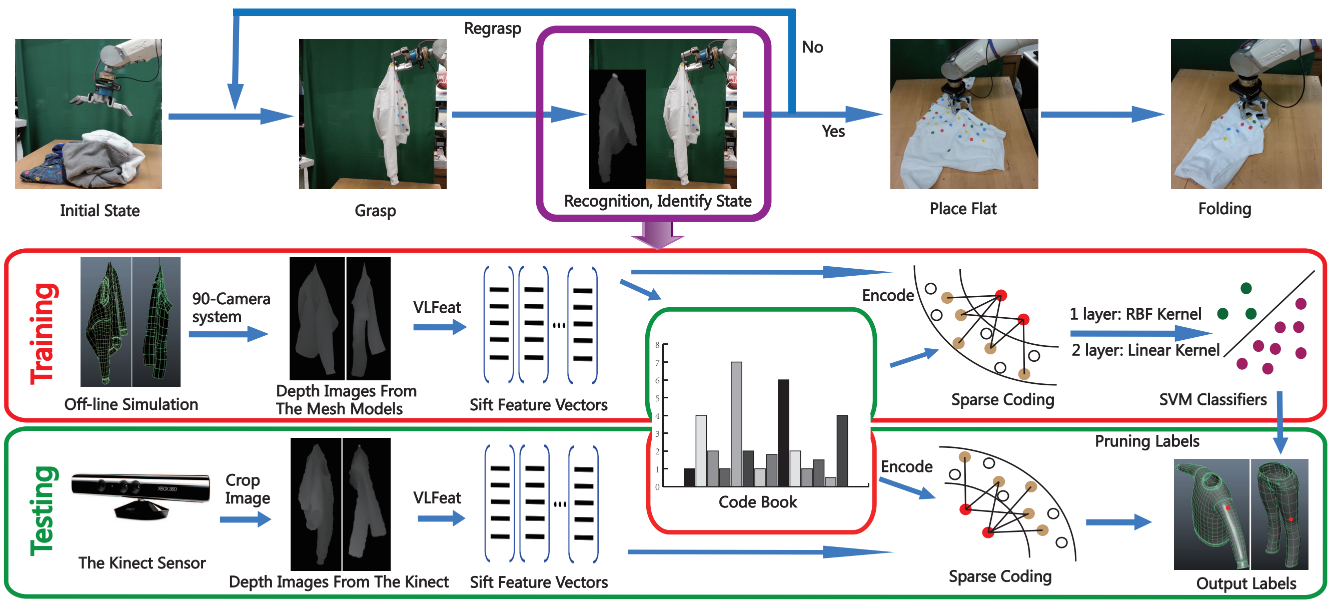

Fig. 2. Overview of our proposed pipeline for manipulation of deformable objects. Top Row: The entire pipeline of dexterous manipulation of deformable objects, which contains five phases. In this paper, we are focusing on the phase of recognition and identification of deformable object category and pose, as highlighted in purple rectangle. If the recognition and identification are not successful, the robot will regrasp the object and repeat the same process again for recognition and identification. Bottom Two Rows: In training flow (in red rectangle), we apply off-line simulation of the deformable objects such as mesh models of sweater and jeans in different poses. By using sparse coding and dictionary learning on depth images, we build up a codebook for recognition. In testing flow (in green rectangle), we capture depth images by the Kinect sensor and use the precomputed codebook to predict the categories and poses of deformable objects.

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, which may fail when the clothes shape is uninformative. A series of works on clothes pose recognition were done by Y. Kita, et. al [6], [8], [7]. Their work demonstrated the ability to identify the pose of the clothes by registration to pre-recorded template images. Without showing more statistical results, it is not clear that the proposed algorithm works on a random grasping point over the entire target clothing.

We formulate our pose estimation problem as an image classification task aiming at depth images of the target garment. We employ the idea of bag-of-feature (BOF) in our method. There are several noticeable works done in the past years [9] [5] [10]. In addition to that, Yang et al. [23] improved the SPM by generalizing vector quantization to sparse coding followed by spatial max-pooling and a linear-kernel SVM. The proposed method outperformed a few previous methods with a relatively high speed. In our approach, we employed Yang's idea with modifications to some specific cases.

III. A Database For Deformable Object Recognition

Figure 2 shows an overview of our pipeline for dexterous manipulation of deformable objects and the focus of this paper (in purple rectangle) which is visual recognition of deformable objects. There are two phases in the recognition. In the first phase, we train on a previously computed database of deformed objects. From that we build a recognition codebook that can be used in the second phase to first recognize the category and then the pose of an object through a two-layer classifier that we have developed. The framework is discussed in detail below.

A. Simulating Deformable Objects

We have developed an off-line simulation process whose results can be used to predict poses of deformable objects. The off-line simulation is time efficient and more accurate compared with acquiring data via sensors from real objects. In the off-line simulation, we use a few well-defined garment mesh models such as sweaters, jeans, and shorts, etc. Figure 1 shows a few of our current garment models rendered in *Maya* software.

B. Generating Training Exemplars

In *Maya*, virtual cameras can be set up to capture depth images of the rendered object. In the real world, a garment can be picked up by a robot with various view points from the camera. So it is meaningful to build up a geodesic dome with cameras placed on it that capture the images from various view points. Our previous work [4] has shown that depth images can help to recognize object categories via feature descriptors and clustering algorithms. Also, we use 3D depth images instead of 2D color images which rely on texture rather than the shape. In our off-line simulation, we set up a group of 90 cameras on a geodesic dome with different view points and positions.

In our experiments, the depth images were captured via the 90 cameras. The workflow of getting the depth images can be summarized as:

1. For each input garment model, a set of $20-50$ grasping points is predefined in terms of the garment categories.

2. For each of the grasping points, a simulation of draping under gravity is carried out and rendered when the garment achieves a stable state (e.g. no shaking).
3. The 90-cameras system will capture depth images of the model from different view points.

IV. Estimating poses Of Deformable Objects

The goal of building a deformable object database is to help in recognizing the category and the pose of a deformable object. In our research scenario, the robot may randomly grasp and pick up a garment without any prior knowledge. After the grasping, we are interested in identifying the category of the garment first and then the position of the grasping point, which are both very important for further actions such as regrasping and folding. From a technical point of view, given a set of features computed from a vision source, (e.g. a depth camera) we would like to index into the deformable object database via feature space, to identify the pose of a targeted deformable object. The problem can be then formulated as an image classification problem. Given a set of depth images captured from a single object, we are interested in identifying the category and grasping point.

A. Feature Extraction

Our method uses the SIFT descriptor which has some outstanding properties, such as rotation invariance. Unlike the approach of using the descriptor in [12], we applied Dense SIFT over our depth images using VLFeat [19]. We also crop the original depth images in terms of the position of the object. This will guarantee that the object always stays in the center of the image, which insures that the spatial pyramid pooling, described below, achieves a better performance. In addition, we discard those sample positions of descriptors that not on the object area for fast computation and encoding.

B. Generating and Learning Feature Signatures

1) Sparse Coding: There are two important components in the BOF model: dictionary learning and feature quantization. The goal of the BOF model is to find a new feature space that has a better ability to represent the training features. Therefore, how to build a *codebook* is crucial in the BOF model. We define a set of N SIFT descriptor as $\{\mathbf{X}\}^N$, where $\mathbf{X}$ is a sample descriptor in the depth image. The problem is then formulated as the following cost function:

$$\min_{V} \sum_{1}^{N} \min_{i=1\ldots k} \|\mathbf{X}_i - \mathbf{v}_i\|^2 \tag{1}$$

where $\mathbf{V} = [\mathbf{v}_{1,\ldots,}\mathbf{v}_k]^{\mathrm{T}}$ is the final k clustering vectors and $\mathbf{v}_{1,\ldots,}\mathbf{v}_k$ are the codewords. We can rewrite the cost function (1) into the following matrix factorization:

$$\min_{W,V} \sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{w}_i\mathbf{V}\|^2 \tag{2}$$

where $\mathbf{W} = [\mathbf{w}_{1,\ldots,}\mathbf{w}_N]^{\mathrm{T}}$ is the weight vector for each of the clusters which will be solved. Traditional method usually applies constraints such that $Cardinality(\mathbf{w}) = 1$ and $|\mathbf{w}_i| = 1$ which lead to vector quantization method. This method only allows feature vectors of one sample be assigned to one "word". Sometimes, it is too restrictive that the *codebook* may have a coarse reconstruction of the original feature vectors. In our scenario, it is highly likely that a depth image from the current view point is in between two or more predefined view points. So it may be inaccurate to reconstruct the current sample via only one word in the *codebook*. In order to improve this, we apply sparse coding method. In the work of Lee et al. [11], it is stated that if another $L1$ norm was added to the cost function (1), the result yields sparsity property, which is called sparse coding. Equation (3) shows the sparse coding equation after we add the $L1$ norm term.

$$\min_{\mathbf{W},\mathbf{V}} \sum_{1}^{N} \|\mathbf{X}_i - \mathbf{w}_i\mathbf{V}\|^2 + \lambda\,|\mathbf{w}_i| \tag{3}$$

In sparse coding, there is more than one non-zero number in $\mathbf{w}_i$, meaning the feature vector can be reconstructed via several "words" in the dictionary. In our problem, the pose of the garment is continuous whereas the training poses are discrete (currently we have defined 90 poses for each grasping point). It is highly likely that the pose of one input garment is in between several training poses. So sparse coding scheme will reduce the errors in reconstruction of the input feature vectors. Since the training task involves large amounts of data, we also employ an online training algorithm that dynamically adjusts the model with less memory consumption (see [13], [14] for details on this method).

2) Max pooling and Linear SVM: In our recognition pipeline, we are interested in a fast classification speed. After the training feature vectors are quantized, we apply a SVM to learn the model. Though the non-linear SVM is more accurate in most of the cases, it brings a high training cost of $O(n^3)$ and a storage cost of $O(n^2)$. According to Yang's [23] work, spatial max-pooling combined with linear-SVM [2] for classification yields an optimal solution for a large data set. After sparse coding on the feature vector, we apply spatial pyramid construction to the feature vector to preserve spatial information. Let $\mathbf{W}'$ be the output weight vector after sparse coding and spatial pyramid construction. The max-pooling function $\mathcal{F}$ is defined on each column of $\mathbf{W}'$. Then the weight vector $\mathbf{r}$ for one depth image can be computed by $\mathcal{F}(\mathbf{W}')$ where $\mathbf{r} = [r_1 \ldots r_J]$. The function $\mathcal{F}$ computes the max weight among each column that is defined as:

$$r_j = \max\{\left|w'_{1j}\right|, \left|w'_{2j}\right| \ldots \left|w'_{Nj}\right|\} \tag{4}$$

According to the observation in [23], spatial max-pooling on sparse coding obtains a high classification accuracy because of less quantization error in sparse coding, robustness in max-pooling to local translation, and sparsity in the image patch. This strategy is only applied to the second layer of the classification which will be described below.

C. Defining Deformable poses

Our approach is to use a large simulated dataset of the garments that have been picked up and hung by gravity to

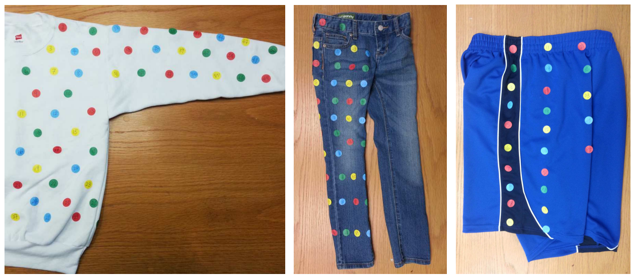

Fig. 3. LEFT: A real sweater with 50 labels pasted on it which are corresponding to the predefined 50 labels in the figure 4. MIDDLE: A real pair of jeans with 40 labels pasted on it. RIGHT: A real pair of shorts with 25 labels pasted on it.

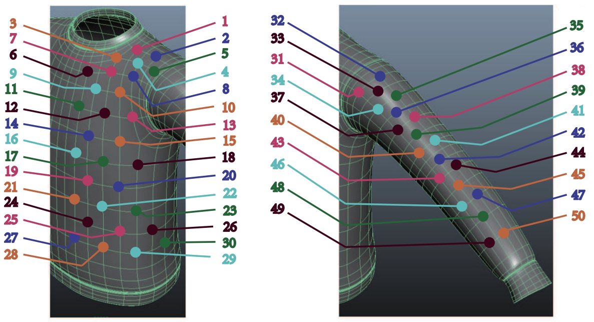

Fig. 4. 50 predefined grasping points within the region of interest on the sweater mesh model. All the grasping points will be picked up once and simulated individually.

predict some unknown poses. For every pick up (grasping point), a pose is defined. Specifically, we predefine a set of grasping points on the garment mesh in terms of structural complexity of the garment. At the same time, we can define the region of interest by eliminating those redundant grasping points based on symmetric geometry of the garment. For example, a sweater, which is symmetric between front and back, left and right so that a reasonable set of grasping points should be within one fourth of the entire surface area. Figure 3 shows color coding label pads on real garments within the region of interest.

Considering the sweater model as an example, given a region of interest, we predefine 50 grasping points on the mesh empirically. These points will be treated as "constraints" in the *Maya* simulation, whereas in real experiments, there are hanging points. For each of the predefined points, we simulate the draping effect in *Maya* and a set of 90 depth images are captured as the training data of this particular grasping point. A sample of the predefined points on the sweater model is shown in Figure 4. We also apply the same predefined points on the real sweater, jeans, and shorts which are shown in Figure 3. The real garments we use are very similar to the mesh models we simulated in *Maya*.

D. Two-layer Classifier

We use a two-layer hierarchical classifier whose structure is shown in Figure 5. For both layers, we first extract features and build up the codebook. Then by applying the sparse coding, we encode features using the codebook. Finally, we use the SVM to classify the garment category and pose. Figure 2 also shows the described work flow.

On the first layer (in red shade), each of the input depth images vote for the garment category and the dominant one yields the final category. On the second layer (in green shade), each input depth image will give an output – a grasping point label. For each grasping point, we may have n_d depth images so that we will have n_d predicted labels correspondingly. Here we apply a label pruning algorithm, which iteratively removes predicted labels that are far from the current mean of all labels, and then outputs one label from a set of labels. Details are described in Algorithm 1.

The only difference between the two layers is that on the first layer we applied a Radial Basis Kernel function (RBF) SVM while the Linear Kernel function SVM was used on the second layer. The RBF kernel is more accurate but relatively slow, so we apply it to the first layer where less classes exist. The Linear kernel is applied to the second layer which usually has 20 to 50 classes.

Algorithm 1 Label Pruning Algorithm

```
Input:
    A predicted label set $\mathbf{\Omega} = \{\mathbf{L_1}, \mathbf{L_2}...\mathbf{L_{n_d}}\}$
    Geometry table $T$ contains 3D coordinates of each label
Output:
    One predicted label $\mathbf{L_p}$ for the current pose
 1: $n = ||\mathbf{\Omega}||$
 2: $center1 \leftarrow 0, center2 \leftarrow 0$
 3: while 1 do
 4:     $center1 \leftarrow \text{mean}(\mathbf{\Omega})$
 5:     $dist\_table \leftarrow \text{pdist}(center1, \mathbf{\Omega})$
 6:     $\mathbf{\Omega} \leftarrow \mathbf{\Omega} - \{5 farthest\_points\}$
 7:     $center2 \leftarrow \text{mean}(\mathbf{\Omega})$
 8:     if $||center1 - center2|| < \xi$ then
 9:         break
10:     end if
11:     if $||\mathbf{\Omega}|| < 0.5 \times n$ then
12:         break
13:     end if
14: end while
15: $dist\_table \leftarrow \text{pdist}\ (center2, T)$
16: $\mathbf{L_p}$ = find label with $\{min(dist\_table)\}$
17: return $\mathbf{L_p}$
```

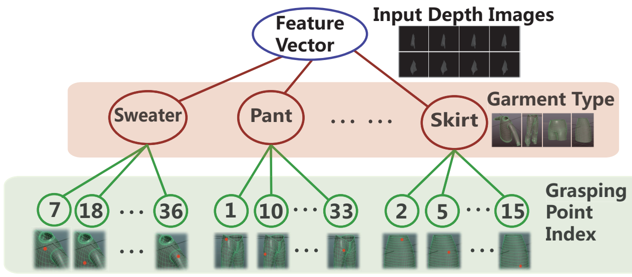

Fig. 5. Structure of the two-layer classifier. The first layer predicts the categories of the garment candidates using an SVM with an RBF kernel whereas the second layer predicts the grasping point label using an SVM with a linear kernel.

V. EXPERIMENTAL RESULTS

A. Test Using Simulation Data

This test focuses on the performance of the second layer of the classifier using simulation data. We have one data sample for one grasping point label. Therefore, each time we take one entire sample of a grasping point label out and test it on the classifier which is trained by the rest of the grasping point

		Sweater	Jeans	Shorts
Height Group	# of Cam	Rank 1	Rank 1	Rank 1
Group 1	9	6.06	6.12	7.19
Group 2	18	4.93	5.12	5.67
Group 3	36	**4.52**	**4.44**	**5.48**
Group 4	18	4.84	5.06	6.29
Group 5	9	5.38	6.05	7.29
Group $2,3,4$	72	5.02	5.09	5.81
All	90	5.79	5.73	6.76

TABLE I. Ranking table of the garment models (sweater, jeans, and shorts) using simulation data from *Maya*. The training images are divided into 7 different groups by height and tested individually. We can see that the rank 1 for sweater and jeans is between 4 and 5 on average. It means the estimated grasping points is very close to the take-out label according to the distribution of the grasping point labels shown in Figure 4. The rank 1 for shorts is a little bit higher which is because the depth images of the shorts are less informative. We can also see that height group 3 always achieves the best recognition rank among all groups.

label samples. If the output labels are close to the take-out label, then the simulation data can be considered continuous and our algorithm should perform well on similar real depth images from a range sensor such as the Kinect.

Taking the sweater model as an example, which has 50 grasping points, each time we select a different target class (a set of depth images from one grasping point) for testing and the rest of the classes (the remaining 49 classes) for training. All depth images can be divided into 5 groups based on their view perspective heights as individual training sets, which are labeled from 1 to 5 from top to bottom. We also add group 2+3+4 and all groups to the training sets. Since we take out the entire testing class, we expect that the estimated grasping point is close to the testing grasping point label. One way to evaluate it is to use the distance ranking as a metric. For example, in Figure 4, if we take out the $17th$ grasping point, the $14th$, $15th$, $16th$, $18th$, $19th$, and $20th$ grasping points can be considered as the closest six points in a sorted order by distance to the $17th$ point. If the $14th$ point is identified as the output, the rank is 1.

We apply this strategy to the simulation data which has three categories, each of which has $25-50$ predefined grasping points. The ranks for all camera groups, and several group combinations are shown in Table I. We can see that the best rank for sweater and jeans are between 4 and 5. This is because on average, there are $4-5$ labels with similar distances to the take-out label, which are considered as the closest label sets. The best rank for shorts is a little bit higher because the depth images of its deformable poses are not as informative as those for sweater and jeans. In real experiments, we also observed that in terms of camera view points of the garment, selecting depth images from certain group(s) as the training data yields better recognition results.

	Garment Categories		
Height Group	Sweater	Jeans	Shorts
Group 3	75.79%	63.33%	84.71 %
Group $2,3,4$	**79.61%**	62.83%	90.76 %
All	73.77%	**72.73%**	**91.40%**

TABLE II. Accuracy table of the garment models (sweater, jeans, and shorts) using depth images from the Kinect sensor. We can see that using all depth images from simulation for training yields better classification results.

B. Test Using Depth Images From the Kinect

Our next experiment is to predict the grasping point of a garment from the depth images captured by a real sensor. We use the Microsoft Kinect as the vision sensor. We used the same (or very similar) garments to our models in off-line simulation. We manually labeled the garments by using color coding label pads, shown in Figure 3, with comparison to the pre-labeled simulated models shown in Figure 4. Then for each of the labels (label pads), we hung and rotated the garments about the grasping point vertically and took 150 to 200 depth images with more than 2 rotating circles.

Table II shows the accuracy of the category classification. (The first layer in Figure 5) In this experiment, we treat each grasping point as an individual test group. Each of the depth images in a test group votes for a garment category. If more than 60% of the depth images vote for one category, we take the result as an output judgment. Otherwise, we discard the current grasping point and move to another grasping point. On average, we discard 20% of the grasping points. The results in Table II are calculated using this strategy. In real experiments, this can be done by regrasping the garment and repeating the whole recognition process. Currently we have three categories which are sweater, jeans, and shorts. Among the depth images taken from various poses and view points from a garment, we also notice that there exists a few, that even for human observers, it is hard to identify the category.

Figure 6 shows some output results of our algorithm on single depth images. The top row is the color images captured by the Kinect sensor with different grasping points. The second row is the depth images, which are captured by the Kinect sensor as well. The third row shows the simulation results of garment mesh models in *Maya* hanging by the ground truth grasping points with similar view perspectives. The fourth row shows the simulation results of garment mesh models in *Maya* hanging by the predicted grasping points with similar view perspectives. We demonstrate both the ground truth and the predicted results to show that our approach can always identify the grasping point with the same or similar appearance from database. The two numbers below the fourth row are the ground truth and predicted grasping point labels (in parenthesis). The number below followed by "cm" is the actually distance between the two labels on the garments.

The last column of the sweater and jeans samples in Figure 6 is one failure example. Comparing the ground truth with the predicted results from database, we can observe that the appearance of the predicted results looks more similar to the real garments. It is because the material property of the real garment is different from the property applied in the off-line simulation, which causes varieties in deformation. In future, we plan to apply different material properties to the garment mesh models in the off-line simulation [18].

In Figure 7, we apply different training sets (height group 3, height group 2+3+4, and all) to test depth images. The X-coordinate of each point in Figure 7 represents the tolerance distance in cm whereas the Y-coordinate represents the

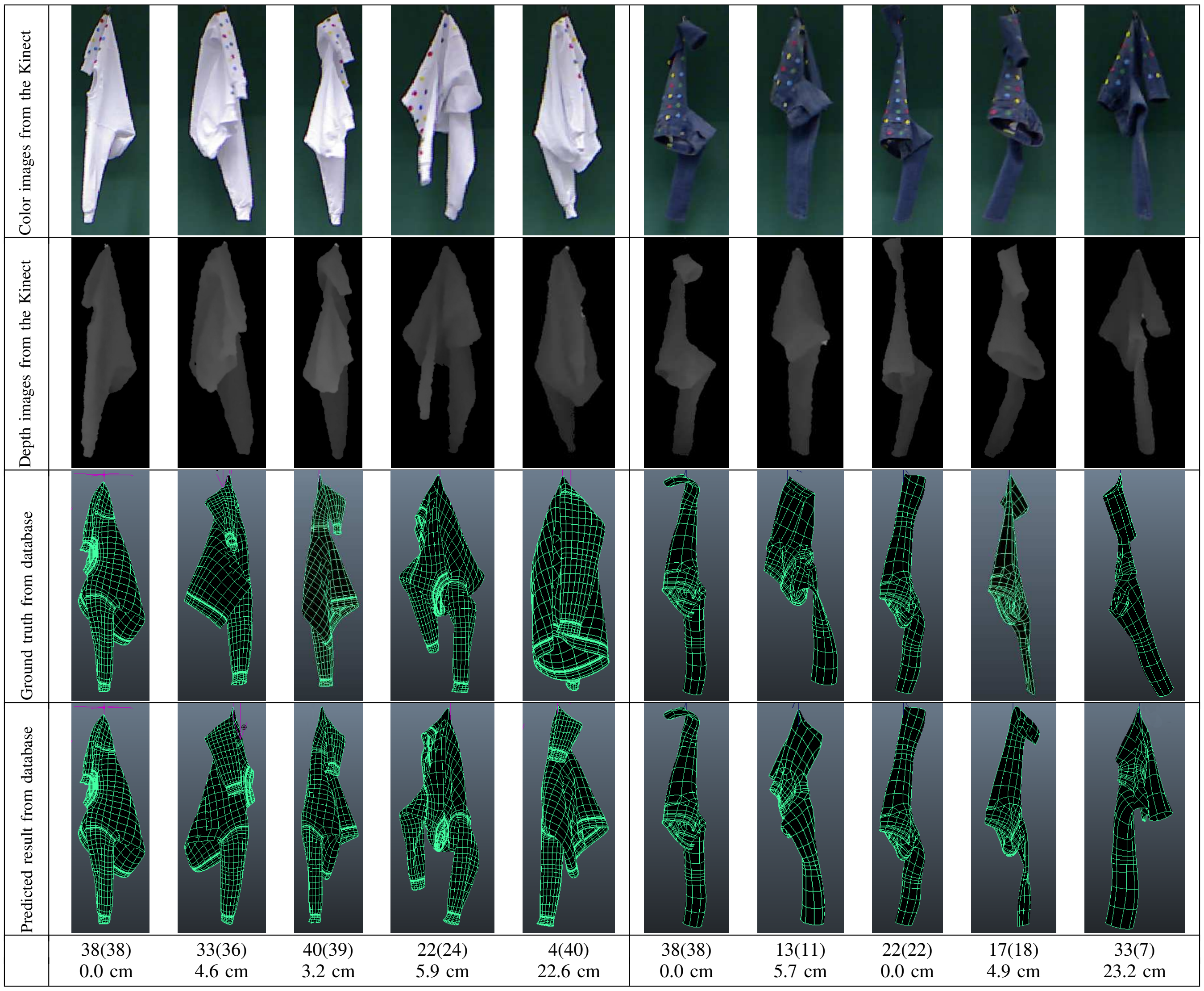

Fig. 6. Sample results from the experiments. The left 5 columns are the sweater while the right 5 columns are the jeans. The top row is the color images captured by the Kinect sensor with different grasping points. The second row is the depth images captured by the Kinect sensor which are in the same pose corresponding to their color images in the first row. The depth images shown are after preprocessing such as filtering, etc. The third and fourth rows show the simulation results of garment mesh models in *Maya* hanging by the ground truth grasping points and predicted grasping points, respectively. The two numbers below the fourth row are the ground truth grasping point labels and predicted labels (in parenthesis). For example, 33(36) means the ground truth grasping point label is 33 whereas the predicted grasping point label is 36. The numbers followed by "cm" are the estimated distances between the ground truth and the predicted labels on the garments. For example, 33(36) and 4.6 cm indicate that the distance between predefined grasping point label 33 and 36 is 4.6 cm. The last column of the sweater and jeans samples is one failure output example of our approach. We can see that the appearance hanging by the predicted grasping points is more similar to the real clothing compared with hanging by the ground truth grasping point.

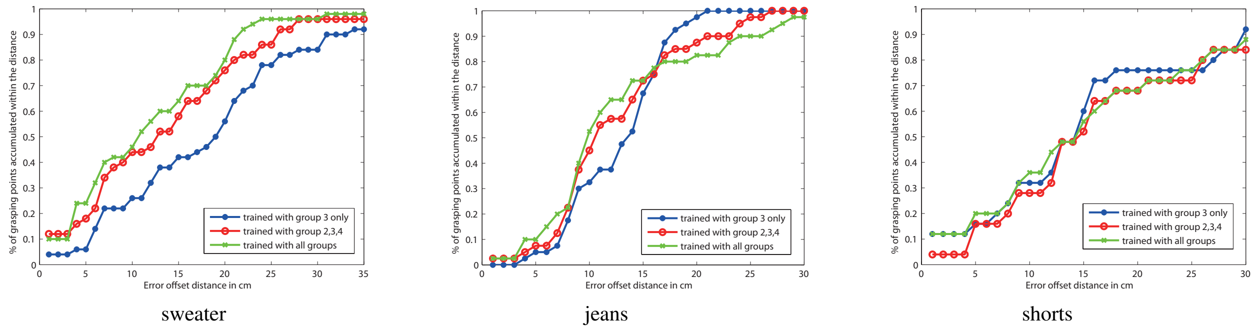

Fig. 7. Accuracy plot for the garment candidates with different training height groups. The X-Axis is the error offset distance measured in cm. The Y-Axis is the percentage of grasping point accumulated within the distance that corresponding to the X value. LEFT: The garment candidate is a sweater that shown in Fig 3. The maximum distance between the two grasping points on the sweater is around 70 cm. MIDDLE: The garment candidate is a pair of jeans. The maximum distance between the two grasping points on the jeans is around 65 cm. RIGHT: The garment candidate is a shorts. The maximum distance between the two grasping points on the shorts is around 51 cm.

accumulated grasping point labels within this distance over the entire garment. For example, one point (10.7, 0.5) on the green curve (trained with all groups) in the plot of the sweater is interpreted as: if we set 10.7 cm as the tolerance distance, 50% of the predefined grasping point labels are recognized correctly. In our case, there are 50 predefined grasping point labels on the sweater mesh model so 25 labels are considered as correct. We observe that in our experiments, training with all depth images achieves slightly better results. We also observe that within 15 cm tolerance distance, around 70% of grasping points can be identified. We believe this accuracy is sufficient for regrasping and folding tasks.

C. Implementation on a Robot

We implemented a grasping algorithm on a Staubli arm and a Barrett hand that picked up a real garment and applied our algorithm to recognize its category and pose. We tested our algorithm on real sweater and jeans through 8 times robot trials of picking up and hanging clothing. Table III shows the results of the the two-layer classifier in the robot experiments. We can see that our algorithm can always predict the grasping point within a short distance from the ground truth. Figure 8 shows snapshots of the process of the recognition pipeline using our approach which contains initial state, grasping and picking up, and recognition from different view points. A video of our experimental results is available at: `http://www.cs.columbia.edu/~yli/DeformObjDepth.html`.

	sweater				jeans			
#	1	2	3	4	5	6	7	8
Cat	S	S	S	S	J	J	J	J
Dist	3.2	15.7	9.1	3.0	7.6	7.4	15.1	10.8

TABLE III. Results of the robot experiments. Totally we have 8 trials. First we classified the category (Cat) for sweater (S) and jeans (J). Then we provide the distance (Dist) in cm between the predicted grasping point and the ground truth.

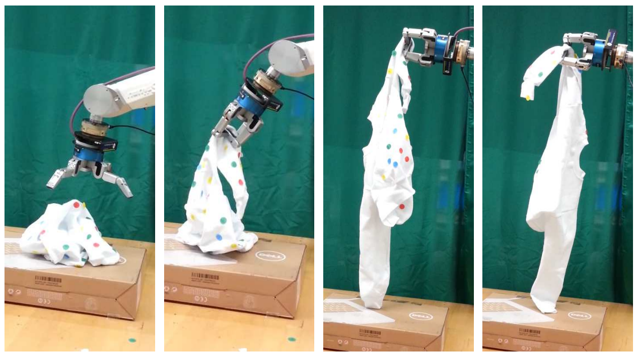

Fig. 8. A Staubli arm grasps a real garment, picks it up, hangs it under gravity, rotate it, and recognizes its category and pose using a Barrett hand. From left to right: initial state, grasping and picking up, recognizing from one view point, rotating and recognizing from another view point.

VI. Conclusion

The entire pipeline of dexterous manipulation of deformable objects includes grasping, recognition, regrasping, placing flat, and folding deformable clothing objects. In this paper, we focus on the recognition. We have shown a framework for recognizing the categories and the poses (grasping points) of a deformable object. We first built up a deformable object database with a set of depth images from off-line simulation as the training data. Then sparse coding was applied to better quantize feature vectors and construct a codebook. We also proposed a two-layer classifier for our classification task. We then tested the performance of the classifier using the off-line simulation data. Meanwhile, we took real depth images via the Microsoft Kinect for testing as well. Both testing results have shown that our proposed framework worked well for estimating the categories and the pose (grasping point) of the deformable object.

Acknowledgments: We would like to thank Prof. E. Grinspun, Prof. C. Zheng, Dr. A. Reiter, J. Weisz, and A. Garg for many insightful discussions. This work was supported by NSF grant 1217904.

References

[1] M. Cusumano-Towner, A. Singh, S. Miller, J. F. OBrien, and P. Abbeel. Bringing clothing into desired configurations with limited perception. In *Proc. ICRA*, 2011.

[2] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[3] P. F Felzenszwalb and D. Ramanan R. B Girshick, D. McAllester. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32:1627–1645, 2010.

[4] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia grasp database. *Proc. ICRA*, 2009.

[5] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *ICCV*, 2005.

[6] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita. Clothes handling based on recognition by strategic observation. In *Humanoid Robots*, 2011.

[7] Y. Kita and N. Kita. A model-driven method of estimating the state of clothes for manipulating it. In *Proc. WACV*, 2002.

[8] Y. Kita, T. Ueshiba, E-S Neo, and N. Kita. Clothes state recognition using 3d observed data. In *Proc. ICRA*, 2011.

[9] Fei-Fei. L and P. Perona. A bayesian hierarchical model for learning natural scene categories. *CVPR*, 2005.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006.

[11] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. *Proc. NIPS*, pages 801–808, 2007.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91 – 110, 2004.

[13] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *Proc. ICML*, 2009.

[14] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19 – 60, 2010.

[15] S. Miller, J. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *IJRR*, 2012.

[16] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. *CVPR*, 2011.

[17] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *Proc. ICRA*, 2013.

[18] N. Umetani, D. M. Kaufman, T. Igarashi, and E. Grinspun. Sensitive Couture for Interactive Garment Editing and Modeling. *ACM Transactions on Graphics*, 30(4), Aug 2011.

[19] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008. http://www.vlfeat.org/.

[20] P-C Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbbeel. Perception for the manipulation of socks. *Proc. IROS*, 2011.

[21] B. Willimon, S. Birchfield, and I. Walker. Classification of clothing using interactive perception. In *Proc. ICRA*, 2011.

[22] B. Willimon, I. Walker, and S. Birchfield. A new approach to clothing classification using mid-level layers. In *Proc. ICRA*, 2013.

[23] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching uisng sparse coding for image classification. *CVPR*, 2009.