On-line learning of temporal state models for flexible objects

N. Bergström, C H. Ek, D. Kragic
CVAP/CSC,
Royal Institute of Technology (KTH) Stockholm, Sweden
{nbergst,chek,dani}@kth.se

Y. Yamakawa, T. Senoo, M. Ishikawa
Graduate School of Information Science and Technology,
University of Tokyo, Tokyo, Japan
{Yuji_Yamakawa,Taku_Seno}@ipc.i.u-tokyo.ac.jp
Masatoshi_Ishikawa@ipc.i.u-tokyo.ac.jp

***Abstract*—State estimation and control are intimately related processes in robot handling of flexible and articulated objects. While for rigid objects, we can generate a CAD model beforehand and a state estimation boils down to estimation of pose or velocity of the object, in case of flexible and articulated objects, such as a cloth, the representation of the object's state is heavily dependent on the task and execution. For example, when folding a cloth, the representation will mainly depend on the way the folding is executed.**

In this paper, we address the problem of learning a temporal object model from observations generated during task execution. We use the case of dynamic cloth folding as a proof-of-concept for our methodology. In cloth folding, the most important information is contained in the temporal structure of the data requiring appropriate representation of the observations, fast state estimation and a suitable prediction mechanism.

Our approach is realized through efficient implementation of feature extraction and a generative process model, exploiting recent hardware advances in conjunction with principled probabilistic models. The model is capable of representing the temporal structure of the data and it is robust to noise in the observations. We present results exploiting our model to classify the success of a folding action.

I. INTRODUCTION

Robots assisting humans in natural environments need to be capable of manipulating complex objects: objects that are articulated or flexible and can be folded or bent. Enabling manipulation of such objects requires modeling of state-space that is more complex that the state-space for rigid objects. One example is origami folding where a piece of paper is transformed into a variety of different shapes. One may model the whole process as a sequence of folds and flips. A similar idea applies to household tasks such as laundry folding. The important scientific question is how to model these sequence so that it suitably describes the possible states of an objects and facilitates robot control. One way of addressing the problem is to use a parametrized model. In the very recent work presented in [17], the authors parametrize different garments by picking out a minimal set of parameters consisting of defining points on the contour, such that the desired range of shapes can be described.

We instead develop a non-parametric approach where we learn the sequence of deformations that define a certain action. In this way, we do not need to create a model manually for a given object, but can rather learn the model from observations. Furthermore, we do not have to define a set of possible actions beforehand. The challenge here is that it is desirable to learn from as few observations as possible. With the proposed model we show that the state space can be kept very small. It is therefore possible to populate that space, and thus learn, from a limited set of observations. Our approach integrates state modeling and prediction of flexible objects. We demonstrate the validity of the method in both *static* scenarios, where the time it takes to perform the manipulation is not an issue, and in *dynamic* scenarios, where the timing must be taken into account. We demonstrate the static scenarios through folding of clothing items, and also apply the method to the dynamic folding scenario presented in [21] to demonstrate the usefulness of the method in scenarios where a high processing speed is essential.

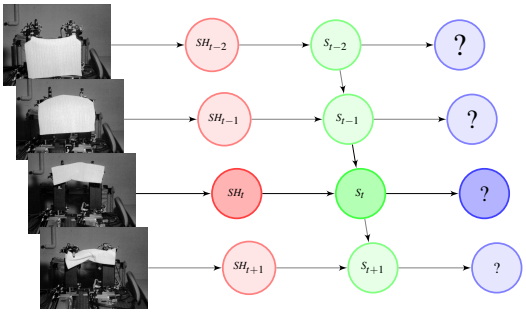


Fig. 1. An overview of the approach: from a video stream, features *SH* are extracted at each image frame. At time t, SH_t together with the previous state S_{t-1} generate a new state S_t. A robot can at any time query the model about its current state. The more observations are available, the more certain the state estimate is. A detailed description of the model is given in Fig. 2.

The main contributions of our work are:

- Methods for efficient extraction of visual features based on an object's shape.
- An approach for classification of a task sequence that continuously integrates new observations and allows for a decision at any given time step in that sequence.
- A fully generative model for state transitions.

The paper is structured as follows: In Sec. III we detail related work. In Sec. IV the method using Shape Contexts is described, and Sec. V details parts of the implementation. In Sec. VI the method is evaluated and the results are discussed. Sec. VII summarizes the paper.

II. SYSTEM OVERVIEW

Several important scenarios in robotic applications are characterized by intricate temporal structures of state changes. An example being the alterations of state induced by the deformation applied to a shirt during folding. These characteristics can be very complicated with respect to both

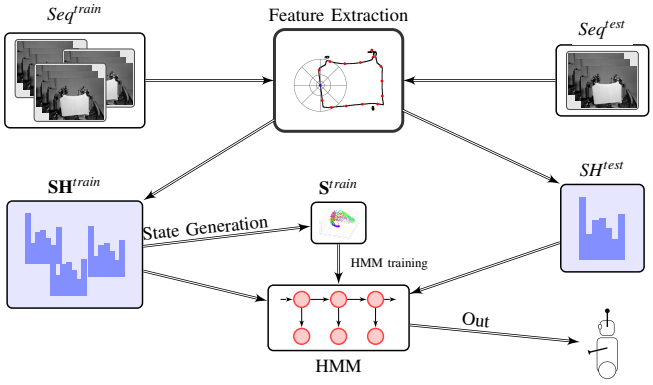


Fig. 2. An overview of the system that follows two paths. In the training stage, a set of sequences, where tasks are performed on an object, are used to generate a set of characteristic states $\mathbf{S}^{train}$ given extracted features $\mathbf{SH}^{train}$. These states are then used for training Hidden Markov Models (HMM) for different actions or tasks. In the execution stage, new observations are continuously processed and fed to the HMM:s, which gives the possibility to observe the state of the object at any time during task execution.

time and space changing at a rapid pace. This makes modeling such data very demanding as it requires detailed, often high-dimensional parameterizations which requires significant computational resources. In this paper we are motivated by the notion that by encoding just a rough temporal structure we can achieve a trade-off between complexity of representation and descriptive power. Reducing the complexity in the state representation allows to focus on the modeling of the temporal signature where we believe a significant portion of the interesting information is retained.

As an example portraying these characteristics, albeit in the extreme, we will look at the scenario of modeling a rapidly folding cloth. This example shows all the challenging characteristics: there is no obvious notion of state and the temporal alterations is happening very rapidly. In this paper, we present a data driven and non-parametric approach to building a model of this scenario. Specifically, by learning the notion of state from data we remove the constraint of a semantically meaningful representation. We exploit this to find a small number of states. The states are in isolation not informative but due to the small number it allows for efficient temporal modeling. Further, the small number of states allows for application of a non-parametric temporal model removing the need to make further assumptions about the transitional characteristics of states.

The central motivation behind the method is that we are observing a *continuous sequence* of object deformation. This means that between each observation, changes in the object are subtile. These changes are not easy to capture with an image based descriptor solely. However, by using a robust descriptor and instead exploiting the temporal aspect of the data by modeling the sequence we get much more descriptive power. An additional requirement concerns the applicability in dynamic scenarios. Here processing speed is of the essence, so the descriptor and modeling should be computationally cheap.

The purpose of the system is to model the state of an object when provided with a video sequence showing the deformation of the object due to some action $a \in \mathscr{A}$. Fig. 2 shows an overview of the proposed system which consists of two separate paths: One training stage and one test stage. A set of video sequences is used as training data, which are generated from observations of different actions applied to the object. The images are processed for features which are used as a basis for generating object *states*, i.e. typical deformations of the object when applied with actions in $\mathscr{A}$. These are then used together with the features to build one temporal model of the sequences for each action.

In the test stage frames from a video sequence are continuously processed for features, which are then used in the temporal model. The robot can at any point query the system for its belief about its current state, something that can than be used for e.g. classification or control.

III. Related Work

Modeling and manipulation of deformable and articulated objects is a natural requirement for a service robot. A popular application in this field of research is manipulation of clothing items. One aspect of this research is to estimate the *state* of the object. By letting the robot hold a garment with one hand Kita *et al*. [14][15] tried to estimate its state by comparing a 3D point cloud of the object to simulated models representing different states of that garment. By knowing the state they could select a second part to grasp and by that spreading the garment. Another work where the authors seek the configuration of clothes is [11]. Similar to this work they exploit observations from the temporal sequence that occur when regrasping the cloth. However, they make these observations when the object has reached a static state, whereas we integrate new observations as they become available.

Another aspect concerns modeling of deformable objects. A popular approach is to represent objects with a number of key points, and model the interactions between these points [8]. This has been used in our previous work for robot motion generation for cloth folding [21].

Planning is another important aspect of manipulation. In [4] the authors addresses robotic folding of clothes. The robot uses the geometry of the object to plan a sequence of folds to reach a final configuration. In [1] Balkcom *et al*. presented work on robotic origami folding. Here the robot is presented with a pattern detailing where the folds on the paper should be done. Given this, the task of the robot is to create a plan to execute these folds. This is a difficult task, but even when the folding becomes more complex, it is like [4] still of the static type. In dynamic scenarios, we cannot plan how the object will evolve. Rather, once the motion is underway we need to observe the object in order to make a *prediction* of its evolution. This makes the dynamic scenario significantly more challenging to model, and requires a suitable prediction model which is difficult to design due to the high number of degrees of freedom of a flexible object.

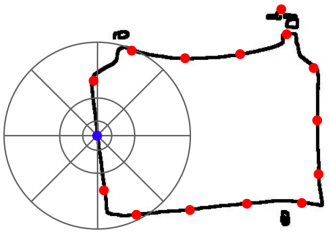


Fig. 3. The shape context for the blue point is created by locating remaining selected points along the contour (red) with respect to the blue point in a log-polar grid, represented by the circle in the figure. The bins in this grid constitute the histogram that is the shape context. For visualization purposes, the outer rings of the grid have been left out.

In [19] the robot spreads a towel by finding and grasping two adjacent corners in the image using vision and two robotic manipulators. The two hands are in constant contact with the cloth, something that cannot always be expected. In our work we instead explore objects in free motion. In [13], the authors presented a high speed dynamic manipulation task consisting of rotating a pizza using a pizza spade as manipulator by controlling its the rotation and translation in depth. The movements of the manipulator is controlled using visual feedback with markers on the pizza. While markers greatly simplify tracking at a high frame rate, using them is not plausible in many real world scenarios.

IV. Method

In the sections below we provide a detailed description of the method, but we first give an overview in relation to Fig. 2. During the training stage a set of video sequences $\{Seq^{train}\}$ are used, which are generated from different actions. For each image in each sequence feature vectors $SH_{t,s}^{train}$ are generated to produce the set $\mathbf{SH}^{train} \triangleq \{SH_{t,s}^{train}\}$. These feature vectors are used to produce a number of states $\mathbf{S}^{train}$. In the following step one Hidden Markov Model (HMM) is trained for each action. $\mathbf{S}^{train}$ serve as basis for the states in these HMM:s.

Features SH_t^{test} are extracted from the test sequence Seq^{test}. These features correspond to observations in the HMM:s, which model the likelihood of the object being in a certain state at a given time step, for a given action. A robot can retrieve information from this system by regarding the likelihood of the sequence having been generated from one of the actions, and how likely the current state estimate of the object is.

A. Feature Vector Extraction

The state of an object is a global property, i.e. the objects deformation can only be described by relating all parts of the objects to each other. By recognizing this, we utilize the *Shape Context* descriptor to represent the image observations [18]. This descriptor works on an object's contour and parametrizes the contour by creating a histogram of distances and orientations between one point on the contour to the rest of the points, see Fig. 3. Contours are extracted with a slightly modified version of the Canny edge detector [9] for the GPU. The difference is that the hysteresis step is disregarded. This reduces edge quality, but in our setting the difference was not significant. The shape context descriptor has been used for many purposes, mainly for matching and recognition [3], [18], but also in a manipulation context by generating grasping points [7]. These works take advantage of the fact that shape contexts can capture a large intra-class variability. In this work we deal with deformable objects such that smooth frame-by-frame transitions will be observed. Nonetheless, depending on the object's initial state the evolution through time will be different, something that shape contexts are able to capture.

We select M key points from the object's contour, which is extracted as explained in Sec. IV. The shape context histogram is divided into r_n radial, and α_n angular bins. An edge image at time t in sequence s is consequently represented with a set of shape contexts $SC_{t,s}$, which holds one shape context corresponding to each of the key points around the contour.

During training, a set of shape contexts for one image, $SC_{t,s}$, is generated for each training image, producing the set $\mathbf{SC}^{train}$. Rather than using these directly as feature vectors, we search for a more compact representation in order to remove correspondence problems in matching. In [18] the authors introduced *Shapemes*. These are canonical base shapes that are formed by vector quantization on the shape contexts using k-means (see Fig. 5). The idea is that key points in close proximity will have very similar shape context, and can be replaced by the same shapeme. By assigning the shape context of each point to the closest shapeme, and creating a *Shapeme Histogram*, we get a much more compact descriptor while retaining most of the descriptive power. Hence matching can be done by comparing shapemes rather than the expensive combinatorial matching done in [3].

Algorithm 1 gives an overview of the feature vector extraction. Details of some of the functions can be found in Sec. V. After training, each image will be described by a shapeme histogram $SH_{t,s}$. The set of shapeme histograms **SH**, created from all training images, form the foundation when training the model.

B. State Generation

In our work there are no predetermined classes when using shape contexts for classification as opposed to previous work utilizing the same feature. Instead, we want to create our own classes, or *states*, from the training set **SH**. Fig. 4(a)

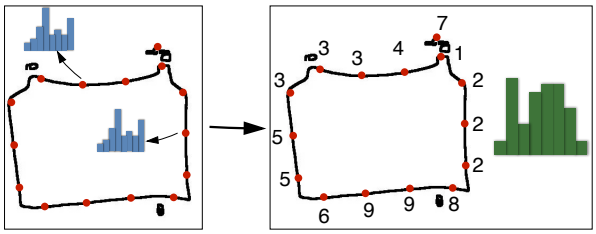
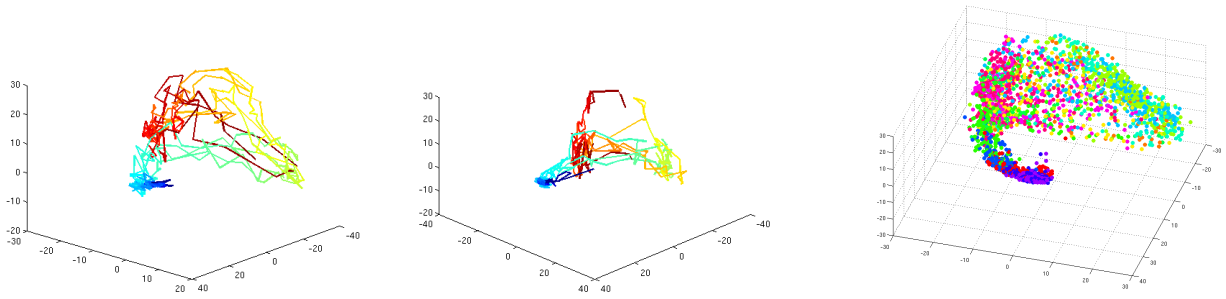


Fig. 5. Shape contexts are first created for each key point on the contour (left figure). After clustering all shape contexts, each key point are assigned to one of these clusters, creating a *shapeme histogram* (right figure).



(a) State transitions for three successful sequences (left) and three failed ones (right) of the first three PCA-components (Principal Component Analysis) of **SH**. The sequences progress from blue to dark red

(b) The plot shows the clustering of the three first PCA-components of **SH**.

Fig. 4. Illustrations of the state space.

shows the evolution of a number of sequences from the first three PCA-components of **SH**. This plot reveals that in the beginning of each sequence, all examples are confined to a restricted volume, and then spread as the object evolves through time. We cluster this space, referred to as the *state space*, to generate the states.

The clustering is done by training an R-component Gaussian Mixture Model (GMM), $p(x) = \sum_{i=1}^{R} \pi_i N(x|\mu_i, \Sigma_i)$, using the Expectation-Maximization-algorithm. The GMM is trained over the first Q PCA-components of **SH**. Since we in this stage are interested in modeling the different states the cloth might be in, all images are used regardless of which action that produced the sequence.

An example of a resulting clustering can be seen in Fig. 4(b). The GMM models the states of the cloth which, at any given instance in a sequence, is computed as the most likely component of the GMM given a shapeme histogram $SH_{t,s}$:

$$S_{t,s} = \text{State}(SH_{t,s}) = \underset{i}{\text{argmax}}\ \pi_i N_i(SH_{t,s}|\mu_i, \Sigma_i), \quad (1)$$

C. Modeling using HMM

Now that each time step in each training sequence is assigned to a state, we train a model over these sequences. Given the temporal nature of the problem, we choose an HMM as our model. An HMM defines a random process that produces data sequences ξ in a way such that the data in each time step is generated from one of a number of states **S**. It is modeled with parameters $\lambda^a = \{q^a, T^a, O\}$. O models observation probabilities $O_i = p(x_t|S_t = i)$, i.e. the probability in time step t of observing x given that state $S_t = i$. O is shared over all actions. T^a models transition probabilities $T^a_{j,i} = p^a(S_{t+1} = j|S_t = i)$ for action a, i.e. the probability of transitioning to state j in the next time step given the current state $S_t = i$. q^a denotes the prior probabilities, i.e. $p^a(S_0 = i)$ for action a.

From Sec. IV-B we have already computed the probabilities O, i.e. if we know which GMM-component that generated the observation, we have

$$O_i = p(SH_{t,s}|S_t = i) = \pi_i N_i(SH_{t,s}|\mu_i, \Sigma_i). \quad (2)$$

Since we have computed $S_{t,s} = \text{State}(SH_{t,s})$, we can compute T^a:

$$T^a_{j,i} = p^a(S_{t+1} = j|S_t = i) = \frac{\sum_{r,s}[S_{r,s} = i \wedge S_{r+1,s} = j]}{\sum_{r,s}[S_{r,s} = i]}. \quad (3)$$

Here $s \in \text{Seq}_{train}$, $r \in 1..N-1$, and $[X]$ is an indicator function

$$[X] = 1 \text{ if } X \text{ is true, } 0 \text{ otherwise.} \quad (4)$$

For each action a we generate an HMM, λ^a. Considering the amount of data that we have (see Sec. VI), we do not have to make any assumptions about the transitions, so a non-parametric estimation of T^a is preferable. q^a can be computed in the same way as T^a, there is however not enough data to make this estimation properly, so we set q^a to be a uniform prior. When creating the state space in Sec. IV-B, we do not make any distinction between sequences from different actions, since we want to model the whole state space of the cloth, so when creating this space we include examples from all actions. Therefore the difference between different actions will largely be the transitions they make through the state space.

For classification scenarios we want to decide whether a new sequence s' is generated from action a. We compare the probability of the sequence being produced by the HMM:s, and use maximum likelihood to select which one:

$$class(s') = \underset{a}{\text{argmax}}\ P(s'|\lambda^a). \quad (5)$$

The probability $P(s'|\lambda^a)$ is computed with the forward algorithm. Although the experiments in this paper perform the classification for the entire sequence, an additional benefit of using a temporal model is that it is possible to read out probabilities from the HMM after any given time step. This means that it is possible to make the classification at an earlier stage, but with potentially reduced certainty.

An additional benefit of using a generative model is that we can synthesize state transitions which could be linked to motor commands for control in manipulation settings.

Algorithm 1 Feature Extraction

for $s \in \text{Seq}_{train}$ **do**
  for $t = 1$ to N **do**
    $Contour_{t,s} \leftarrow$ extractContours($Image_{t,s}$)
    $P_{t,s} \leftarrow$ getKeyPoints($Contour_{t,s}, M$)
    $SC_{t,s} \leftarrow$ getShapeContext($P_{t,s}$)
SC $\leftarrow \{SC_{t,s} | \forall t, \forall s\}$
$Shapemes \leftarrow$ k-means(**SC**, K)
for each $SC_{t,s}$ in **SC** **do**
  for $i = 1$ to M **do**
    $SH_{t,s}[\ getClosestShapeme(SC_{t,s}(i))\ ] \leftarrow 1$
SH $\leftarrow \{SH_{t,s} | \forall t, \forall s\}$

Algorithm 2 $P \leftarrow$ getKeyPoints($Contour, M$)

1) Parallel $b \in B$: $s^b \leftarrow$ countEdgepixels($Contour$)
2) CPU: $cs^0 \leftarrow 0$
for $b \in B$ **do**
  $cs^b = cs^{b-1} + s^b$
3) Parallel $b \in B$:
  $P_{all}[cs^{b-1} : cs^b] \leftarrow$ getEdgepixels($Contour$)
4) $P \leftarrow$ sample(P_{all}, M)

Algorithm 3 $SC \leftarrow$ getShapeContext(P)

Require: $SC[a,b] = 0$
$d_x = \log \alpha |p - P(x)|$, $\theta_x =$ angularBin$(p, P(x))$
atomicAdd($SC[d_x, \theta_x]$,1)

V. Implementation Details

Using the graphics processing unit (GPU) significantly speeds up certain types of processing, in particular the image processing, in which tasks are often easily parallelizable, like computing gradients for the edge detection. Others are easier to implement sequentially on the CPU. Here we provide implementation details for some specific problems that were solved on the GPU to speed up processing.

a) Key Point Sampling: Given an edge image, we need to sample the contour in M places. These should be evenly distributed over the contour. In a sequential implementation the first step would have been to search for contour points in the image, generate a list of contour points by tracing the contours and then evenly sampling this list. The problem with a GPU implementation is that the GPU processes small blocks (16x16 pixels in our case) of the image separately, and there is no way to transfer information between these blocks in the same execution. Therefore we cannot know how many key points to sample from each block. We solve this by running the sampling in several execution steps according to Algorithm 2. Step 1) is done using parallel reduction on each block, and step 3) can be done by first sorting the edge pixels within a block, and then copying the number of edge pixels in that block, to the final list P_{tot}. Sorting within a block can be done efficiently using *bitonic sort* [2]. Finally step 4) samples this list on the CPU, which is done by randomly picking one point, and then evenly sample another $M-1$ points. The results will slightly differ from the sequential algorithm, but it will produce an evenly distributed sampling.

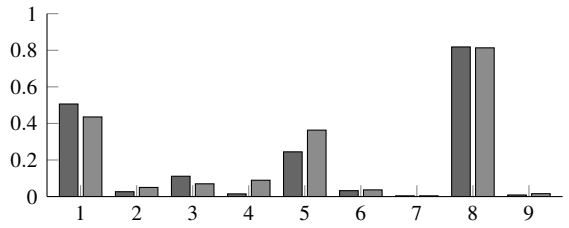


Fig. 7. The distribution over states for the two actions in the static scenario.

b) Shape Context Generation: For generating shape contexts for a set of key points, we let each block compute the shape context for one point p, and each thread in that block is responsible for computing one distance. The details of the shape context kernel can be found in Algorithm 3. Here angularBin(p,q) returns the index for one of six equally spaced angular bins that the point pair p,q is placed in. By using atomicAdd, we ensure that two threads cannot simultaneously increase one specific bin.

VI. Experiments and Evaluation

We run two different experiments showing the validity of our approach. The first experiment is static, describing the folding of a shirt, The second experiment is different, being extremely dynamic, depicting the rapid folding of a piece of thin cloth. The challenges of the two experiments are different, in the first experiment the state transitions are slow but the variance in the different states is large. This will test the descriptive range of the image feature. In the latter experiment the state changes are very subtle which will test the resolution of the state representation. Furthermore, the changes are very rapid testing the resolution of our algorithm and the efficiency of our implementation. The model is trained as described in Sec. IV, and the testing procedure for a sequence is as follows:

1) For each image, detect contours, select key points and compute shape contexts.
2) For each shape context, find closest shapeme and construct **SH**, i.e. the sequence of feature vectors for the test images.
3) Project the vectors on the Q first PCA-components from the training.
4) Run **SH** on $\lambda^a, \forall a$ using the forward algorithm.

A. Static Scenario

We demonstrate the system in a t-shirt folding scenario using two actions, *correct fold* and *bad fold* (Fig. 6). The execution of both actions are similar: Folding the left side, the right side and then from the top. With the latter action the fold was however done sloppily, producing a result deviating from the expected one. We recorded ten sequences of each action, varying aspects like time to fold, final size of the folded t-shirt, and in the *bad fold* case, different deviations from the *correct fold*. Fig. 7 shows the distributions of states for both actions. These are very similar, indicating that it is

Fig. 6. Examples of instances of the *correct fold* action (top) and the *bad fold* action (bottom).

not about which states the t-shirt takes during the action. This justifies the central belief of this paper, that the information lies in the *transitions* through these states.

For training we randomly selected six sequences of each action, and used the remaining for testing. The procedure was repeated 100 times to get an average. The sequences were between 400 and 600 frames, and we use every fifth frame for training. Between 1000 and 2000 contour points were detected in each image, and $M = 250$ of them were sampled. For Shape Contexts, 10 angular and 6 radial bins were used, and $K = 32$ shapemes were created. We tried different R and Q values, but not so much difference could be observed. For $R = 11, Q = 7$ we got an average classification result of 0.81 for both actions.

Having a temporal model is beneficial in many ways since it also allows for early detection of errors. In the static case the HMM also makes the method time-invariant. One-shot classification, such as classifying each frame using an SVM, would be hard since the data is not aligned. We tried to classify the last image of the sequences (the folded shirt) with an SVM, but this turned out to give poor performance slightly above random. Even if the data would be aligned, we would run into the problem of having too few examples to train from, contrary to the sequence of data for the temporal model as discussed above. In the dynamic scenario the data is in fact aligned due to the same motion repeatedly being produced by the robot. In this case we compare the temporal model with a different SVM-approach as discussed next.

B. Dynamic Scenario

In order to test the system in dynamic scenarios, we use the scenario of [21] in which the robot holds a cloth at its upper rim and grabs the lower rim in the air by swinging the cloth (See Fig. 8). Due to parameters that are hard to control, like initial position and stretching of the cloth, many attempts fail. This results in two actions being used: *successful folding* and *failed folding*. We let the robot discriminate between these cases. The whole folding sequence takes around 400 *ms* which means that the modeling has to be done at high speed.

We conducted experiments using 9 instances of the successful folding, and 18 of the failed folding. Images are captured at 1000 Hz, and we use every third frame for both training and testing. On average between 500 and 1500 contour points were detected in the images, and we sample $M = 128$ from each one of them. For training, 12 sequences, 6 from each action, containing 128 images each were used, which gives $|\mathbf{SC}^{train}| \simeq 200\ 000$ vectors. The vectors were clustered into $K = 32$ shapemes. 12 sequences with 128 images each means that 1536 Shapeme Histograms were used to create the state space. By using $R \leq 13$ we have on average at least 118 vectors per cluster. As in the static case, we repeated this 100 times to get a reliable average.

Some parameters need to be set, mainly concerning the number of key points to be used, the size of the shape context histogram, the number of shapemes, how many PCA-components to use and how many clusters in the final state space. We present results by varying the final number of PCA-components to use, Q, and the number of clusters used in the state space, R. Tab. I shows results from running the classification over a few different parameter settings. As seen from the table, like in the static case the results does not vary significantly for the different parameter settings, indicating robustness of the method. The numbers are taken from sequences classified using all available observations. The somewhat low classification rate indicates that there is overlap between successful and unsuccessful attempts. This is partly due to the single frontal view of the scene not providing enough discriminative information.

Fig. 9 shows two examples of the output of the HMM classification throughout the sequence, averaged over 100 runs. As expected, given the results in Fig. 4(a), in the beginning of the sequence the classification is random, while the benefit of using transitions can be seen after a bit more than half way into the sequence when the classification rate of both successful and failed examples increases.

For reference, to emphasize that the classification is depending on time, we use a bag-of-words approach and train a support vector machine (SVM) as follows: For time steps $1,..,t$ we create a histogram over the states computed as

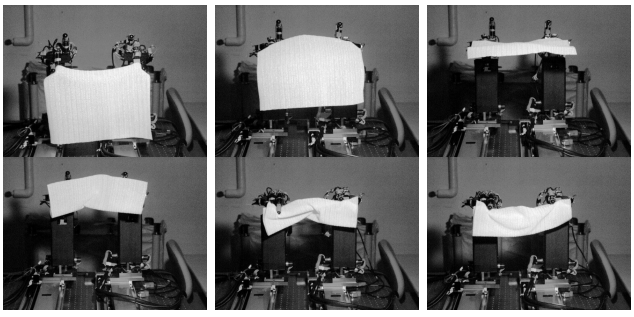


Fig. 8. Frames 0, 100, 200, 300, 400 and 450 of a successful folding.

(**R**, **Q**)	(9,3)	(9,5)	(9,7)	(9,9)	(11,3)	(11,5)	(11,7)	(11,9)	(13,3)	(13,5)	(13,7)	(13,9)
TP	0.72	0.75	0.70	0.67	0.74	0.77	0.70	0.77	0.68	0.72	0.69	0.73
TN	0.70	0.67	0.63	0.67	0.69	0.68	0.69	0.66	0.70	0.72	0.70	0.68

TABLE I

THE TABLE SHOWS THE TRUE POSITIVE (CORRECTLY CLASSIFIED SUCCESSFUL EXAMPLES) AND TRUE NEGATIVE RATES (CORRECTLY CLASSIFIED FAILED EXAMPLES) FROM RUNNING THE CLASSIFICATION OVER A NUMBER OF PARAMETER SETTINGS (R,Q). BEST PERFORMANCE IS ACHIEVED FOR $(R,Q) = (13,5)$ WHICH GIVES A CLASSIFICATION RATE OF 72% FOR BOTH CLASSES.

in Eq. 1 and train the SVM using these. When t is small there is naturally too little information for a good classifier, but even as t grows, the classifier fluctuates around random performance, clearly showing the need for a temporal interpretation of the data. Furthermore, by using an HMM there is natural compensation for differences in time to perform an action, as shown in the static case.

VII. CONCLUSIONS

Robots will need the ability to observe and understand actions that are performed around them. One essential aspect of this is understanding how objects behave. While rigid objects are easily modeled, flexible objects are inherently much more complex, and therefore maybe even more important to understand. Furthermore, some actions that are dynamic in nature, i.e. the those where the temporal aspect cannot be affected by the performer, puts additional requirements on the modeling.

In this work we used folding of objects as scenarios for modeling and evaluating an effective integration of vision based representation for complex objects in synergy with a principled probabilistic method for encoding of temporal tasks. The model describes how the contour of the object evolves through a folding sequence. We showed how this can be used to monitor that the execution is evolving as planned. The system was evaluated in two scenarios - one involving human folding and one involving rapid robot motion. Our model is fully generative and could be coupled with an additional control mechanism in order to generate a more active robot execution. The model also allows us to make continuous predictions over time, accumulating evidence to make a more informed decision.

The next step is to integrate the presented system in the dynamic scenario with an active feedback loop to perform corrective motions improving the ratio of successful attempts.

Acknowledgements: This work was funded by IST-FP7-Collaborative Project-270436 TOMSY, the Swedish Foundation for Strategic Research (SSF) and Scandinavia-Japan Sasakawa Foundation.

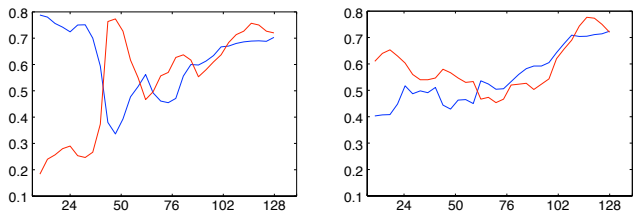


Fig. 9. The plots show the classification rates over time for parameter settings $(Q,R) = (9,3)$ (left), and $(Q,R) = (13,3)$ (right) as read out from the HMM:s, and averaged over 100 runs.

REFERENCES

[1] D. J. Balkcom and M. T. Mason. Robotic origami folding. In The International Journal of Robotics Research, 27(5):613–627, (2008)

[2] K.E. Batcher. "Sorting Networks and their Applications." In Proc. AFIPS Spring Joint Comput. Conf., Vol. 32, 307-314 (1968)

[3] S. Belongie, J. Malik, and J. Puzicha. "Shape matching and object recognition using shape contexts." In IEEE Transactions on Patt. Anal. and Machine Intelligence, 24:509–522, (2002)

[4] J. V. D. Berg, S. Miller, K. Goldberg, and P. Abbeel. "Gravity-based robotic cloth folding." In Proc. WAFR, (2010)

[5] N. Bergström, C.H. Ek, M. Björkman, and D. Kragic. "Scene Understanding through Autonomous Interactive Perception." In International Conference on Computer Vision Systems France, Sept. (2011)

[6] N. Bergström, Y. Yamakawa, T. Senoo, M. Ishikawa. "State Recognition of Deformable Objects Using Shape Context" In The 29th Annual Conference of the Robotics Society of Japan AC1E1-2 (2011)

[7] J. Bohg and D. Kragic. "Learning grasping points with shape context." In Robotics and Autonomous Systems, 58(4):362 – 377, (2010)

[8] D. E. Breen, D. H. House, M. J. Wozny, and D. E. Breen. "Predicting the drape of woven cloth using interacting particles." In Proceedings of SIGGRAPH, pages 365–372, (1994)

[9] J. Canny, "A Computational Approach To Edge Detection." In IEEE Trans. Pattern Anal. and Machine Intelligence, 8(6):679–698, (1986)

[10] NVIDIA CUDA "Compute Unified Device Architecture Programming Guide" NVIDIA: Santa Clara, CA, (2007)

[11] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel. "Bringing clothing into desired configurations with limited perception." In Proceedings of IEEE International Conference on Robotics and Automation, pages 1–8, May (2011)

[12] N. Furukawa, T. Senoo, A. Namiki, and M. Ishikawa. "Dynamic Regrasping Using a High-speed Multifingered Hand and a High-speed Vision System." In IEEE ICRA 2006 pages 181–187 (2006)

[13] M. Higashimori, Y. Omoto, and M. Kaneko. "Non-grasp manipulation of deformable object by using pizza handling mechanism." In IEEE ICRA 2006 120 –125, May (2009).

[14] Y. Kita, T. Ueshiba, E. Neo, and N. Kita. "Clothes state recognition using 3d observed data". In IEEE International Conference on Robotics and Automation, ICRA '09. pages 1220 –1225, may 2009.

[15] Y. Kita, T. Ueshiba, E. Neo, and N. Kita. "A method for handling a specific part of clothing by dual arms." In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 4180 –4185, Oct. (2009).

[16] Y. Luo and R. Duraiswami. "Canny edge detection on nVidia CUDA." In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops 1 –8, june 2008.

[17] S. Miller and M. Fritz and T. Darrell and P. Abbeel. "Parametrized Shape Models For Clothing." In International Conference on Robotics and Automation (ICRA) 2011.

[18] G. Mori, S. Belongie, and J. Malik. "Efficient shape matching using shape contexts." In IEEE Transactions on Pattern Analysis and Machine Intelligence 27(11):1832 –1837, Nov. (2005).

[19] K. Salleh, H. Seki, Y. Kamiya, and M. Hikizu. "Inchworm robot grippers for clothes manipulation." In Artificial Life and Robotics, 12:142–147, (2008).

[20] P.C. Wang, S. Miller, M. Fritz, T. Darrell, P. Abbeel. "Perception for the Manipulation of Socks" In In IEEE/RSJ International Conference on Intelligent Robots and Systems Sept. (2011)

[21] Y. Yamakawa, A. Namiki and M. Ishikawa. "Dynamic Folding of a Cloth by Two High-speed Multifingered Hands." 2010 JSME Conference on Robotics and Mechatronics, 2A1-F05 (2010)

[22] Y. Yamakawa, A. Namiki and M. Ishikawa. "Dynamic Folding of a Cloth by High-speed Multifingered Hand System using Visual Feedback." 28th Ann. Conf. of the Rob. Soc. of Japan, 1O3-6 (2010)