Deformable Contour Tracking using Hidden Markov Models

Chen-Ping Yu, Kyle Brocklehurst, Sitapa Rujikietgumjorn

I. Introduction

Object tracking is useful in many applications, and it is an active research area of computer vision. It is commonly used in human computer interaction and visual surveillance systems. Conventional methods such as background subtraction and mode seeking have been widely used, while many new approaches involving active shape models and graphical models were proposed over the past few years. However, these approaches usually use a pre-defined template for matching, which allows very little deformation.

In this project, we focused on object tracking using contours. Specifically, we attempt to track objects that may deform substantially. Where this can be useful is for image segmentation and general object tracking. We formulate the contour as a Hidden Markov Model, which was proposed by Huang [1]. In each new frame, we allow the states (locations along the boundary) to adapt to new observations of edge detection and NCC score against a patch centered at their location along the previous frame, while constraining smoothness of neighboring transitions as well. To do this, the forward-backward algorithm is used to find the global optimum set of contour landmark points in the new frame.

II. Related Works

A Hidden Markov Model (HMM) is a probabilistic representation that is widely used in speech recognition applications and proved to be a robust method for related recognition systems. It has also been applied to recognize hand gestures to interpret American Sign Language [2], where a user's hands are tracked from skin color, and the tracking process focuses on hand movement through time. A coarse description of hand shape, orientation, and trajectory are inputed to a HMM for recognition.

HMM can also be used to detect the contour of an object [1]. The contour can be detected based on various cues and constraints. In [1], they used foreground and background color, pixel intensity, and edge information as observations for each hidden state of the HMM. For this project, our observations are gradient (Canny edge detection) and normalized cross correlation (NCC) scores only. However, instead of limiting ourselves to an ellipsoid object contour as in [1], we try to adjust the contour to the object edge which gives a more complex representation as the object may change to a very complex shape, such as change of hand gestures. Here, the object contour is represented by its landmark points. The predicted position of a landmark point in a subsequent frame is restricted to be on the normal line to the contour through the landmark's position in the current frame.

III. Methods

We implement an HMM to track object contour based on the formulation from [1], where the hidden states of the HMM are the landmark points along the contour of the object in a new frame. The observations are the gradients (Canny edge detection) and the NCC score (against a patch from the previous state), taken along a 21 pixel long 4-connected path centered on the landmark's location in the previous frame and extending along the normal to the objects contour at that point in the previous frame. Figure 1 shows the details of how HMM is formulated for this object tracking problem.

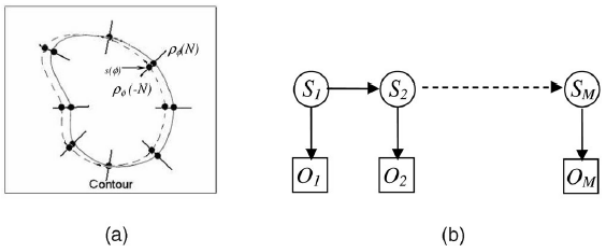

Fig. 1: A) The solid line is the previous contour and the dashed line is the next contour. Normal lines are shown at the landmark points. B) The hidden states S are inferred by the observations O. (image from [1])

Since our goal is to track deformable object contours, we made each state seek a new position between each frame. This enables our method to adapt to the new shape of the object contour. This also makes our method scalable in the number of landmark points, as the boundary of the object is recalculated at each frame and the landmarks are always taken at every fifth boundary pixel. Thus, in frames where the object is smaller we require fewer landmarks points to track and dynamically increase the number of landmark points as the surface area of the object increases.

In our observation matrix, each row represents the pixels along a normal line at one of the contour's landmark points, while the columns represent the states (pixel offsets along that line). To populate this matrix, the value for any position along any normal line is computed from the combination of its NCC and gradient probability, where we want the true state to be similar to the previous frame (NCC) and on the edge of the object (gradient). State transitions are modeled by an overall contour smoothness constraint, which is a Gaussian penalty of $e^{-\beta(x_j - x_i)^2}$ where x are the states on consecutive normal lines. Thus, we are trying to minimize a weighted cost associated with finding the positions along the normals that have the best edge response, look the most like the position in the previous

frame, and are smoothest with respect to the transition being suggested by the landmark points on either side of the one in question. We calculate smoothness as the probability of observing the states in the rows directly above and below the row in question with respect to a gaussian centered at the state (column) at that row. To obtain the global optimal solution, we implemented the Forward-Backward algorithm to propagate the smoothness information round-trip along all states, and then use Dynamic Programming to compute the sequence of state transitions that yields highest likelihood and lowest cost. Forward-Backward Belief Propagation can be implemented using either MAP or MMSE.

$$MAP : m_{i\to j}(x_i) = \max_{x_j}[\phi(x_i)\psi(x_i, x_j)\prod m_{k_i}(x_i)] \quad (1)$$

$$MMSE : m_{i\to j}(x_i) = \sum_{x_j}[\phi(x_i)\psi(x_i, x_j)\prod m_{k_i}(x_i)] \quad (2)$$

The important difference between the two is whether the maximum value from the propagated observations or the sum of the values is used in the iterations. Overall, our method contains the following steps:

1 Manually draw initial contour

2 For every C pixels on the contour, generate the landmark points and its normal lines

3 Obtain hidden state observations with gradient and NCC information, generate observation matrix

4 Employ smoothness constraint into state transition probability

5 Apply Forward-Backward Belief Propagation with MAP or MMSE, followed by Dynamic Programming to find the lowest cost state transition

6 Compute optimal state transitions and adjust landmark points to determine new contour

7 Repeat steps 2-6 for continuous deformable object contour tracking

IV. Experiments and Results

We have implemented the algorithm using MMSE estimation in hope that it is a better way to avoid false positive nearby strong edges. We have tried a few different input videos to test its performance and the output is interesting. The method runs at about 5 seconds per frame (depending on the number of landmark points being tracked). In our experiments, we used every fifth pixel along the contour as a landmark point; the more dense you make this spacing, the more accurate the tracking but the longer it takes to run. We allowed the Forward-Backward algorithm to run for 10 iterations, though it was usually observed to converge at around the fifth iteration. For the gaussian that determines penalties for violating smoothness, we used a constant weighting term beta = 0.01. This indicates how much smoothness matters with respect to aligning with the best edges and strongest correlation. If this value is lower, it allows for greater deformation but may be taken away from the object when another strong edge passes by. If the term is greater, it will help preserve the shape of the object and may help keep it from being misled by nearby edges, but it will not allow for great deformation between subsequent frames, so deformation needs to be slow with respect to frame rate.

Test case 1: simple rotating fist: Figure 2

Fig. 2: Tracking an object with translation and slight deformation such as turning your fist is fairly accurate and can recover even after temporarily losing the contour.

Test case 2: flexing of the fingers: Figure 3

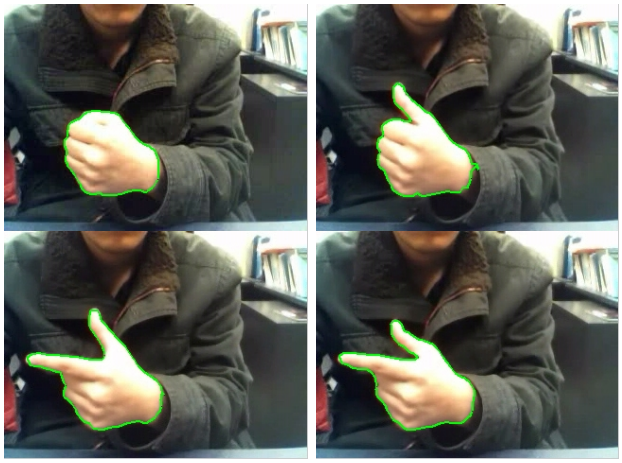

Fig. 3: Our method is capable of tracking complex deformations and maintains accuracy by adjusting the number of landmark points as the surface area increases.

Test case 3: brain tumor segmentation: Figure 4

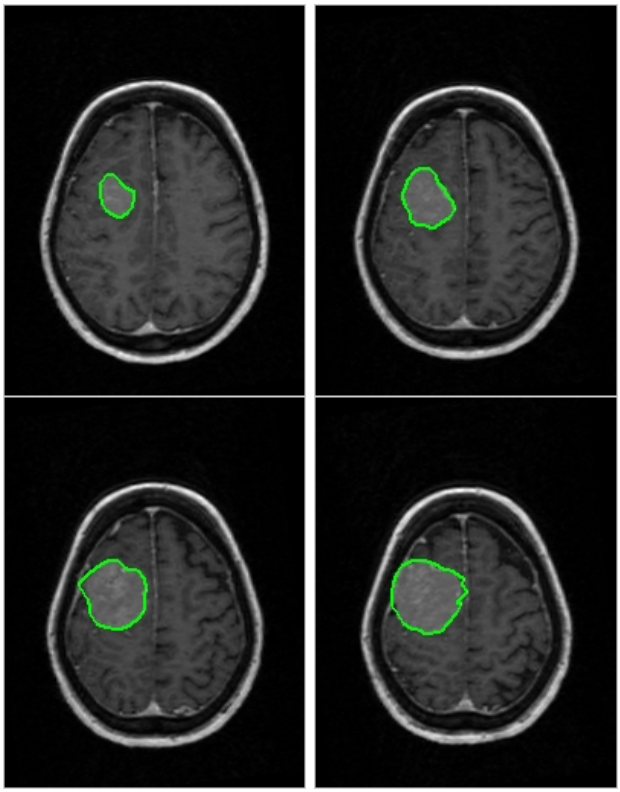

Fig. 4: Another application to HMM deformable contour tracking is for brain tumor segmentation, treating each consecutive slice as continuing video frames.

Test case 4: hand extending all fingers: Figure 5

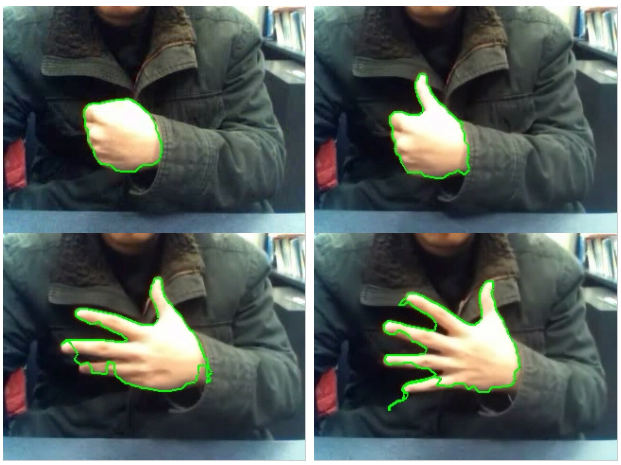

Fig. 5: This shows a failure of our algorithm. The deformation here is too extreme and happening too quickly. This represents an area where either the cost weighting parameters or the similarity and edge measures could be improved.

V. Conclusions and Future Work

We have implemented contour tracking using HMM, and we have added our twist to make the algorithm track deformable object contour. We show our success in tracking objects with translation and deformation due to rotation, perspective, and actual deforming movement of the object itself. Our experiments show promising results, while sometimes tracking can be thrown off by strong edges nearby, but that recovery is possible as the object continues to deform. Much work on exploring the parameter space can be done, changing the way smoothness is weighted and taking more criteria into account than just edge detection and NCC score. In particular, a good extension would be to record a color or intensity histogram for the foreground (inside the contour) and background (outside). Using an observation that attempts to keep appropriate colors inside and exclude others should help avoid many situations where a nearby edge disrupts tracking. Our method could prove useful in many applications, such as tumor segmentation, where one slice of the tumor can be found by seeking assymmetry and could then be segmented in adjacent frames using our method, as well as in sign language recognition [2], where the ability to describe the position and pose of the hands and fingers with more complexity than an approximate ellipse [1] could lead to the recognition of more complex signs and allow for a greater vocabulary.

References

[1] Y. Chen, Y. Rui, and T. S. Huang, "Multicue hmm-ukf for real-time contour tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1525–1529, 2006.

[2] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371 – 1375, 1998.