Recognizing Temporal Trajectories using the Condensation Algorithm

Michael J. Black* **Allan D. Jepson†**

* *Xerox Palo Alto Research Center,* 3333 Coyote Hill Road, Palo Alto, CA 94304

† *Department of Computer Science*, University of Toronto, Toronto, Ont. M5S 1A4

black@parc.xerox.com jepson@vis.toronto.edu

Abstract

The recognition of human gestures in image sequences is an important and challenging problem that enables a host of human-computer interaction applications. This paper describes an incremental recognition strategy that is an extension of the "Condensation" algorithm proposed by Isard and Blake (ECCV'96). Gestures are modeled as temporal trajectories of some estimated parameter over time (in this case velocity). The condensation algorithm is used to incrementally match the gesture models to the input data. The method is demonstrated with an example of an augmented office whiteboard in which a user makes simple hand gestures to grab regions of the board, print them, save them, etc.

1 Introduction

The recognition of human gestures in image sequences is an important and challenging problem that enables a host of human-computer interaction applications. The dominant paradigm involves computing low-level feature information at each frame derived from motion or model matching. The parameters of these low level features evolve over time and form *temporal trajectories*. Recognition is typically performed using these trajectories via Hidden Markov Models (HMM's) or Dynamic Time Warping (DTW).

DTW explicitly matches a stored trajectory with in input trajectory while allowing local deformations of the curves to produce a best match. HMM's typically do not maintain detailed trajectory information but rather break the trajectories up into discrete states that are represented by the mean and covariance of the parameters in that state. One advantage of HMM's is that they provide a probabilistic framework for gesture recognition.

Our goal is to combine the best features of DTW and HMM's; that is we seek a recognition method that can capture the detailed trajectory information as in DTW and yet has a probabilistic framework as in HMM's. Our method probabilistically matches model trajectories to input trajectories in an "on-line" fashion. The proposed method exploits the CONDENSATION (CONDitional dENSity propagATION) algorithm proposed by Isard and Blake in [2] and recently extended in [3]. A variety of parameters must be estimated to match a model trajectory and the input trajectory. The probability distribution over all the match parameters is represented by discrete random samples. This distribution evolves over time as the input data changes. The condensation algorithm uses stochastic dynamics and random sampling techniques to "track" this distribution as it evolves.

In [3], the authors used a very simple temporal model that represented the mean and covariance of the velocity of a tracked object. This is an impoverished temporal model and the authors demonstrated only simple forms of gesture recognition. The method proposed here allows much more powerful temporal models and hence can be used to recognize more complex gestures. In our experiments we use gesture data gathered by a computer vision system that is observing an office whiteboard. We describe a vocabulary of gestures that a user can perform at the whiteboard to extend its functionality. The system could also be used to recognize other sorts of gestures as well as on-line handwriting.

In the following section we describe our extended Condensation algorithm. Section 3 presents the whiteboard application and experimental results.

2 Condensation Algorithm

Our goal is to take a set of M model trajectories $\{\mathbf{m}^{(\mu)}, \mu = 1, \ldots, M\}$ and match them against an input trajectory (see Figure 1). The models are taken to be discretely sampled curves (though they may be continuous as well) with a phase parameter $\phi \in [0, \phi_{\max}]$ representing the current position in the model. The model values at position ϕ are a vector of N values $\mathbf{m}_\phi^{(\mu)} = (m_{\phi,1}^{(\mu)}, \ldots, m_{\phi,N}^{(\mu)})$ where the stored discrete curve is linearly interpolated at phase ϕ. At time t the input trajectory is an observation vector $\mathbf{z}_t = (z_{t,1}, \ldots, z_{t,N})$.

The parameters we need to estimate to match a model to the data are:

μ: an integer indicating which model is being matched,

ϕ: the position (or phase) within the model that aligns the model with the data at time t,

α: an amplitude parameter that is used to scale the model vertically to match the data, and

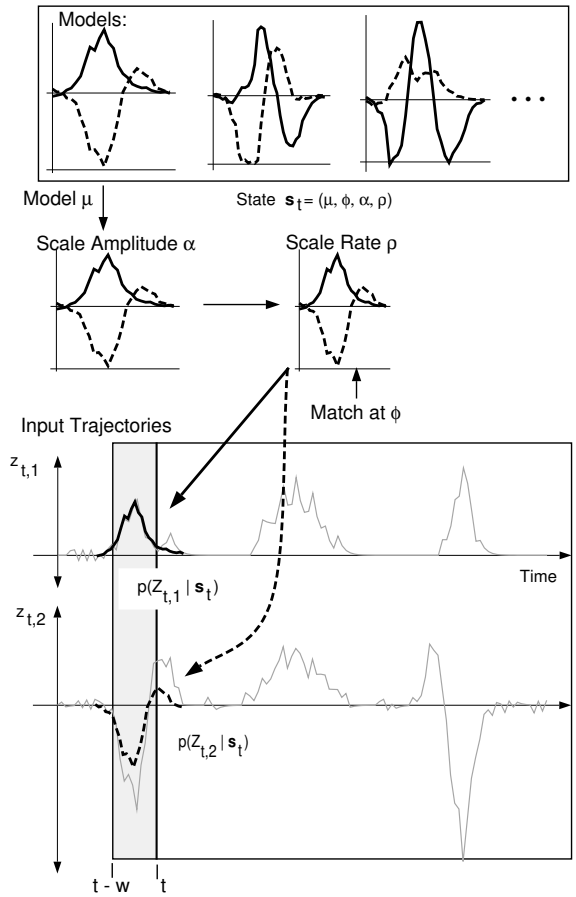

Figure 1: Our goal is to incrementally match M, multiple-parameter, trajectory models to input data.

ρ: a rate parameter that is used to scale the model in the time dimension.

We define a state at time t to be a vector of parameters $\mathbf{s}_t = (\mu, \phi, \alpha, \rho)$.

We would like the find the state $\mathbf{s}_t$ that is most likely to have given rise to the observed data $Z_t = (\mathbf{z}_t, \mathbf{z}_{t-1}, \ldots)$. Let $Z_{t,i} = (z_{t,i}, z_{(t-1),i}, z_{(t-2),i}, \ldots)$ be the vector of observations of a particular trajectory, i, over time. We define the probability of an observation $\mathbf{z}_t$ given the state $\mathbf{s}_t$ as

$$p(\mathbf{z}_t|\mathbf{s}_t) = \Pi_{i=1}^{N} p(Z_{t,i}|\mathbf{s}_t), \tag{1}$$

where

$$p(Z_{t,i}|\mathbf{s}_t) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp \frac{-\sum_{j=0}^{w-1} \left(z_{(t-j),i} - \alpha m^{(\mu)}_{(\phi-\rho j),i}\right)^2}{2\sigma_i(w-1)}$$

and where w is the size of a temporal window backwards in time over which we want the curves to match. The σ_i are estimates of the standard deviation for curve i. Also $\alpha m^{(\mu)}_{(\phi-\rho j),i}$ is simply the the value of trajectory i in model μ interpolated at time $\phi - \rho j$ and scaled by α.

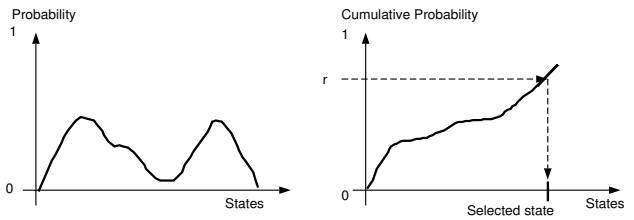

Figure 2: We sample from the distribution over the states by constructing a cumulative probability distribution and sampling it uniformly. Choosing r from a uniform distribution over $[0,\ 1]$ gives us a corresponding state.

Given a definition for $p(\mathbf{z}_t|\mathbf{s}_t)$, we can construct a discrete representation of the entire probability distribution over the possible states. Initially, we sample uniformly form

$$\begin{aligned} \mu &\in [0, \mu_{\max}] \\ \phi &= \frac{1-\sqrt{y}}{\sqrt{y}}, \text{ where } y \in [0,1] \\ \alpha &\in [\alpha_{\min}, \alpha_{\max}] \\ \rho &\in [\rho_{\min}, \rho_{\max}]. \end{aligned}$$

Note that the prediction for the initial phase ϕ is biased towards small values. We then compute the probability of the state $p(\mathbf{z}_t|\mathbf{s}_t)$. We can do this for S samples where we take S on the order of 1000. This gives us a set $\{\mathbf{s}_t^{(n)}, n = 1, \ldots, S\}$ of samples. We normalize these probabilities so that they sum to one, producing weights $\pi_t^{(n)}$

$$\pi_t^{(n)} = \frac{p(\mathbf{z}_t|\mathbf{s}_t^{(n)})}{\sum_{i=1}^{S} p(\mathbf{z}_t|\mathbf{s}_t^{(i)})} \tag{2}$$

The condensation algorithm [2, 3] uses this information (the sample states and their weights) to predict the entire probability distribution at the next time instant. Unlike traditional tracking methods (e.g. Kalman filtering) this approach can deal well with ambiguous data since a distribution of possible matches is propagated in time. The algorithm has three stages (selection, prediction, updating). Below we outline how to construct a new probability distribution with S samples at time t given the the distribution at time $t-1$.

1. Selection: First we choose a state from time $t-1$ *according to the probability distribution* at time $t-1$. That is, we use the current probability distribution over states to choose likely states to predict into the future.

This is done by constructing a cumulative probability distribution using the $\pi_{t-1}^{(n)}$ as illustrated in Figure 2. Let the cumulative probabilities be

$$\begin{aligned} c_{t-1}^{(0)} &= 0 \\ c_{t-1}^{(n)} &= c_{t-1}^{(n-1)} + \pi_{t-1}^{(n)}. \end{aligned}$$

We sample this distribution by uniformly choosing a value, r, between zero and one. We then find the smallest $c_{t-1}^{(n)}$ such that $c_{t-1}^{(n)} > r$. The state $\mathbf{s}_{t-1}^{(n)}$ is then selected for propagation to the next time instant.

With this sampling method, states will be selected according to their estimated probability. To avoid getting trapped in local maxima and to deal with sudden changes in the input data, we randomly choose some fraction of the states to be replaced by random initial guesses (initialized as described above). We typically take these random guesses to be $5 - 10\%$ of the samples.

2. Prediction: Given a state $\mathbf{s}_{t-1}^{(n)}$ we predict the parameters of the new state $\mathbf{s}_t^{(n)}$ at time t to be

$$\begin{aligned} \mu_t &= \mu_{t-1} \\ \phi_t &= \phi_{t-1} + \rho + \mathcal{N}(\sigma_\phi) \\ \alpha_t &= \alpha_{t-1} + \mathcal{N}(\sigma_\alpha) \\ \rho_t &= \rho_{t-1} + \mathcal{N}(\sigma_\rho) \end{aligned}$$

where $\mathcal{N}()$ is a normal distribution and the σ_* represent uncertainty in the prediction. Note that prediction can be viewed as sampling from the probability distribution $p(\mathbf{s}_t^{(n)}|\mathbf{s}_{t-1}^{(n)})$ [3]. For the time being, μ does not change over time; we will extend the method below to allow transition probabilities to take μ to a new state.

During prediction if $\phi_t > \phi_{\max}$ then that model has been recognized and the state is initialized as described above. For the other parameters, we draw samples from $\mathcal{N}$ until the predicted value of the parameter is within its allowed range.

It is interesting to note that the σ_* are used as a tool for locally searching the parameter space. They can be thought of as "diffusion" parameters that blur the probability distribution as it is predicted in time. They provide a way of performing a local search about a state. They also allow local deformations of the trajectories within the moving window of size w.

3. Updating: Given a new state we evaluate the probability, $p(\mathbf{z}_t|\mathbf{s}_t^{(n)})$, that it generated the data at time t using Equation (1). If the likelihood is zero (or below a threshold) then we return to step 2 for a new prediction. We repeat this for a fixed number of tries. If no prediction from $\mathbf{s}_{t-1}^{(n)}$ has sufficient probability then we generate a random initial sample.

After S new states have been generated, we compute the normalized weights, $\pi_t^{(n)}$, using Equation (2) and repeat the process for the next time instant.

Note that the condensation algorithm is a probabilistic hybrid of depth-first and breadth-first search. When no good match to the input data is found, the method resorts to uniform sampling (breadth-first). When the probability mass is centered around a set of parameters, more resources will automatically be spent to explore the neighborhood around these parameters (depth-first). Also, note that a "simulated annealing" method could be used in the search to start with high values of σ_i in Equation (1) and lower them over time.

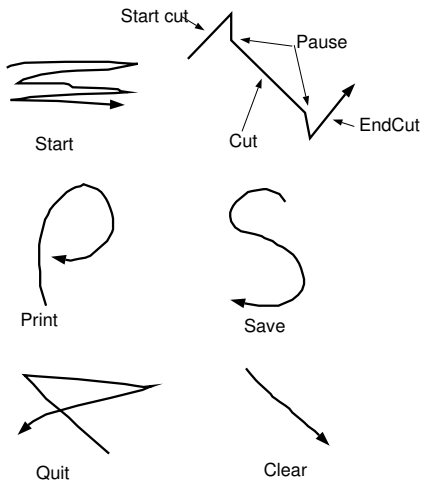

Figure 3: Gestures.

2.1 Finite State Extension

We have also extended the method to allow compound models that are very like Hidden Markov Models in which each state in the HMM is one of our trajectory models. Let $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_i\}$ be a "parent state" which consists of a set of event, or model, types μ_i along with transition probabilities $p(\mu_t|\mathcal{M}, \mu_{t-1}, \phi_{t-1})$ between states. In the prediction step above, μ_t is chosen by sampling from the transition probabilities. When initializing a new random state, we first chose the parent type $\mathcal{M}$ from a uniform distribution. The initial event type μ is defined by the parent. The remaining parameters are chosen as described above.

3 Gesture Models

To test the condensation-based trajectory recognition algorithm we consider the problem of recognizing a set of gestures in the context of an augmented whiteboard. A number of authors have looked at problem of scanning whiteboards at high resolution using mosaicing [6] and interacting with the board by making hand-drawn marks [5]. Here we look at the problem of recognizing dynamic gestures. Isard and Blake [3] used the condensation algorithm to recognize very simple drawing gestures. Our extension to temporal trajectories allows more complex gestures to be recognized.

In our scenario, when the user wants to perform a command, they pick up a gesture "phicon" (or physical icon) [4] that has a distinctive color that makes it easy to locate and track. The motion of the phicon is tracked using a color histogram tracker [1] in real time. Tracking is performed at roughly 30Hz. Since the tracking rate varies slightly, we resample the phicon locations at fixed time instants using lin-

ear interpolation. The horizontal and vertical velocity of the phicon are used for gesture recognition.

We define the following set of gestures which are useful for such a purpose (see Figure 3):

- **Start**: The start gesture tells the system to "pay attention" and start recognizing gestures. This gesture is simply a waving motion similar to what a person might do to get a human's attention.
- **Cut Region:** The cut gesture is used to indicate a region of the whiteboard to be scanned (possibly at higher resolution). This gesture consists of three primitive gestures with pauses in between of arbitrary duration. We refer to the complete cut gesture as a "parent" gesture that is made up of "events".
  - **Cut-On**: The gesture begins with an upside-down "check mark." The end of this *cut-on* gesture marks the upper left corner of the scanning region.
  - **Cut**: The user then moves the phicon in a relatively straight line to the lower right corner of the region.
  - **Cut-Off**: To end the gesture and cut the image region the user makes a right-side-up "check mark".
- **Print**: To send a cut region to the printer, the user makes a gesture like the letter "P".
- **Save**: To save the region to a file, the user makes a gesture like the letter "S".
- **Clear**: A sharp diagonal motion "clears" the current stored whiteboard region. One can think of a cut region as being "stored" in the phicon. The gesture can be thought of as "throwing" the cut region away.
- **Quit**: The user makes a mark like an "X" when they wish to stop the gesture recognition function.
- **Stationary**: In addition to the gesture models we also represent when the phicon is stationary.

To construct models for the gestures, each gesture was performed approximately half a dozen times and the trajectories were saved. For a given gesture the training trajectories were manually aligned and the mean trajectories were computed. A standard deviation from the mean trajectory was also computed for each curve. The trajectory models for each gesture are shown in Figure 4. The initial alignment of the curves could be performed using DTW. In addition to computing just the mean curve, we could compute "Eigen-Curves" as in [7].

In our experiments we take $\mu_{\max} = 9$ (there are nine primitive events), $\alpha_{\max} = 1.3$, $\alpha_{\min} = 0.7$ (we allow a 30% scaling), $\rho_{\max} = 1.3$, $\rho_{\min} = 0.7$ (a 30% temporal scaling). The standard deviations, σ_i, for the model trajectories were taken to be 1.0. Finally, the diffusion parameters

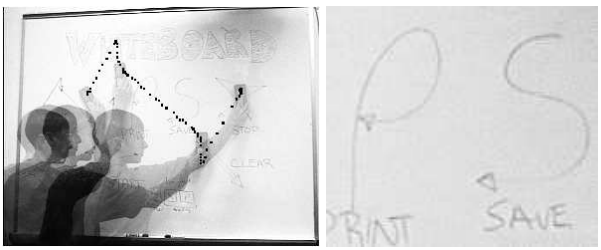

Figure 5: Example of a "Cut" gesture. The user makes a gesture with the phicon. The image on the right is the region that is cut out of the larger image.

were taken to be $\sigma_\rho = 0.01$ $\sigma_\phi = 0.05$ $\sigma_\alpha = 0.01$, and the temporal window was $w = 10$.

Figure 5 illustrates the performance of a "cut" gesture. We use a bright red block as our gesture phicon. The black dots in the figure represent tracked locations of the phicon. The image on the right is the region that is cut out by our algorithm.

Figure 6*a* shows the horizontal and vertical velocities of the phicon as a function of time. This is our input data to which we will incrementally match the gesture models.

The estimated value of the horizontal and vertical velocity is taken to be

$$\mathbf{u}_t = \sum_{n=1}^{S} \pi_t^{(n)} (\alpha \mathbf{m}_\phi^{(\mu)})$$

where $\alpha \mathbf{m}_\phi^{(\mu)}$ is the estimated velocity for sample state $\mathbf{s}_t^{(n)}$. This represents the fit of the models to the data and is shown in Figure 6*b*.

Figure 6*c* shows the probability of each individual event as a function of time. Similarly Figure 6*d* shows the probability for the composite, parent, gestures. The probability of a particular event, μ, is taken to be the sum of the normalized probabilities $\pi_t^{(n)}$ for which $\mu \in \mathbf{s}_t^{(n)}$.

Figures 6*e* and *f* show the probability that the events or parent gestures have completed respectively. This probability is given by

$$p(\mu^*) = \sum_{n=1}^{S} \begin{cases} \pi_t^{(n)} & \text{if } \mu \in \mathbf{s}_t^{(n)} \text{ and } \phi + 1 > \phi_{\max} \\ 0 & \text{otherwise} \end{cases}$$

where a sample is considered completed if the estimated phase ϕ parameter is within one time instant of the maximum phase for that event/parent. The figures show clear spikes when the individual events ("Cut On", "Cut," and "Cut Off") end. Similarly there is a spike when the parent cut gesture ends. If $p(\mu^*) > 0.1$ we consider μ to be recognized.

To actually cut the region (Figure 5) from the original image we must carry along with each state the time at which

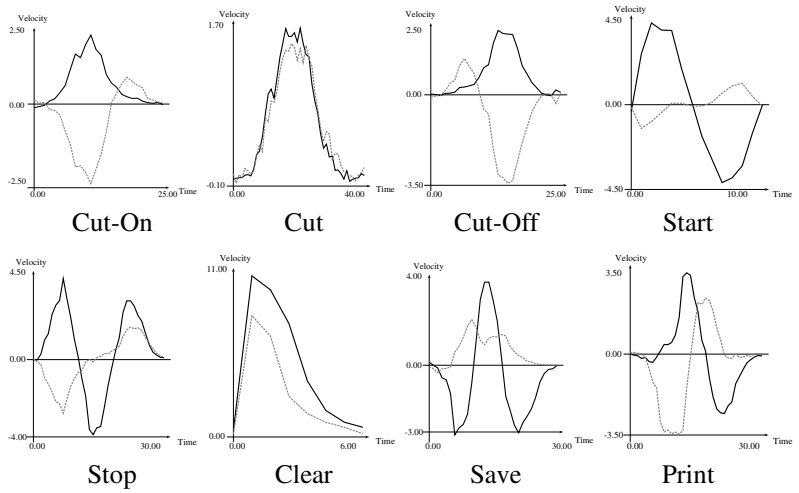

Figure 4: Gesture Models. Temporal trajectories of horizontal (solid) and vertical (broken) velocity.

each primitive event ended. This information is used at the end of the gesture to determine where the stationary mark points are located. The enclosed region of pixels is then extracted.

3.1 Multiple Gesture Experiment

We consider a more complex experiment that involves a series of gestures

Cut – Save – Clear – Cut – Print – Clear.

The input curves represent 850 samples of the horizontal an vertical velocities of the phicon (Figure 7 top). In this sequence, in addition to the actual gestures, the user moves the phicon between the gestures.

There are nine possible event types and six possible gestures. The input data at every frame is matched against these models and the data must be explained by some model. The normalized probabilities in Figure 7 (bottom) show that some of the non-gesture motions receive high normalized probabilities as we would expect. But using the probability for those gestures that actually are completed ($p(\mu^*) > 0.1$) we successfully recognize the completion of the gestures as shown along the bottom of Figure 7.

4 Conclusions

We have described an extension to the Condensation algorithm that performs probabilistic matching of model curves to input curves. This method allows the recognition of more complex gestures than is possible with the standard Condensation algorithm. We have also described an application of the method for gesture recognition for an office whiteboard scanner.

Note that while here we use the condensation algorithm to perform recognition given the gesture trajectories, the algorithm can be extended to actually perform the tracking as well [2, 3].

Currently the method is significantly slower than real time. Blake and Isard, however, have demonstrated real-time versions of a similar algorithm which suggests that our method might be suitable for real-time recognition (with appropriate optimizations).

In our current work, segmented and aligned training data was provided. Transition probabilities were set by hand. While the method has very few parameters, it would be worth exploring how learning techniques could be used to generate the models automatically.

Acknowledgements. We thank Francois Bérard for providing the real-time phicon tracking and Gudrun Socher for discussions about whiteboards and gesture interfaces.

References

[1] J. Coutaz, F. Bérard, and J. Crowley. Coordination of perceptual processes for computer mediated communication. *Int. Conf. on Auto. Face and Gesture Recog.*, pp. 106–111, 1996.

[2] M. Isard and A. Blake. Contour tracking by sotchastic propagation of conditional density. *ECCV-96*, pp. 343–356, Cambridge, UK, 1996.

[3] M. Isard and A. Blake. A mixed-state Condensation tracker with automatic model-switching. *ICCV*, pp. 107–112, Mumbai, India, Jan. 1998.

[4] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. *Proc. of CHI'97*, 1997.

[5] Q. Stafford-Fraser. Brightboard: A video-augmented environment. *Proc. of CHI'96*, 1996.

[6] R. Szeliski. Image mosaicing for tele-reality applications. *IEEE WACV*, pp. 44–53, Sarasota, FL, 1994.

[7] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *ICCV*, pp. 120–127, Mumbai, India, Jan. 1998.

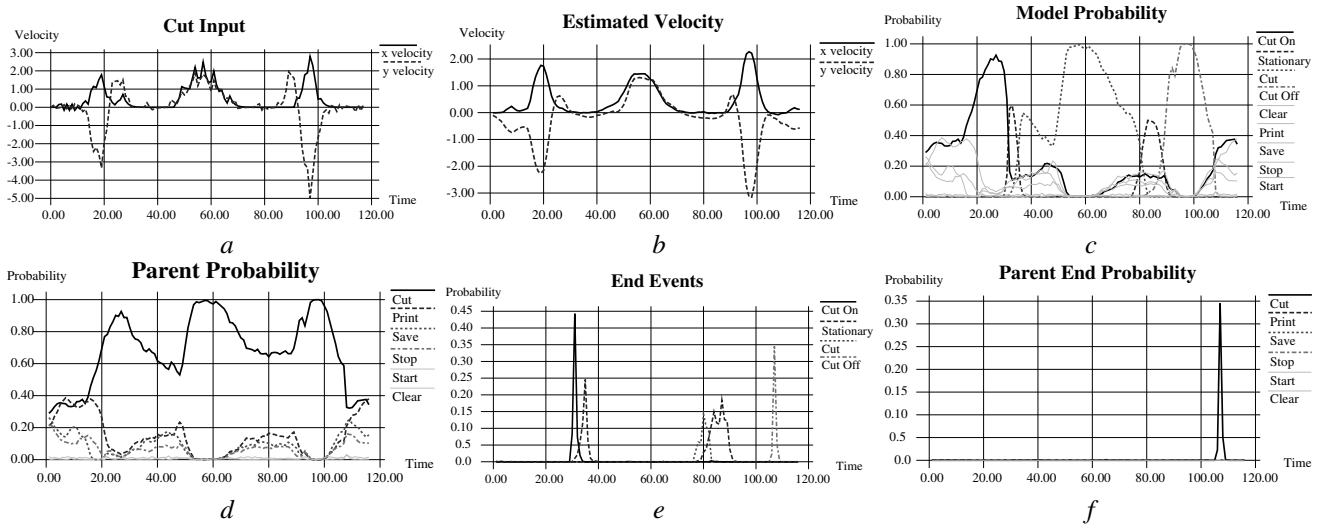

Figure 6: "Cut" gesture. *(a),* Input horizontal and vertical velocity; *(b),* Estimated horizontal and vertical velocity; *(c),* Probability of each event type; *(d),* Probability of each parent gesture type; *(e),* Probability that the current state is the completion of an event; *(f),* Probability that the current state is the completion of a parent.

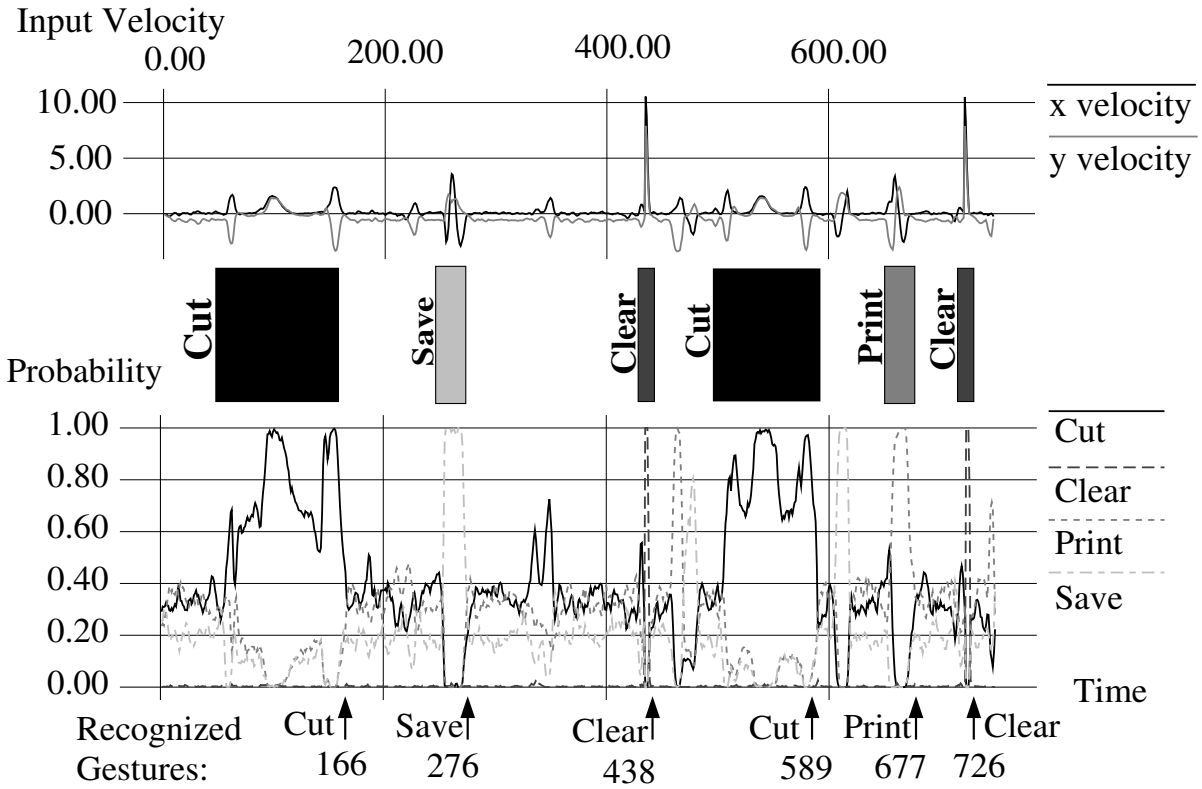

Figure 7: Multiple-gesture experiment. *top:* Input horizontal and vertical velocity; *bottom:* Probability of each parent gesture type. The frame number at which the recognized gesture completes is given.