# Categorization and Learning of Pen Motion using Hidden Markov Models

David Tausky
University of Waterloo
datausky@uwaterloo.ca

Richard Mann
University of Waterloo
mannr@uwaterloo.ca

**Abstract:** In this paper we present a framework for the classification and segmentation of motion data. First, a representation of different two-dimensional motion categories is proposed. Secondly, a system to categorize and segment motion is presented based on hidden Markov models, commonly used in speech recognition. Specifically, this paper combines concepts from both handwriting theory and computational vision to create a system to classify the motion of a pen when writing/drawing. Input to the system consists of online pen stroke data which includes the x, y position and time of each point along the line. Using derived velocity and angle information the system classifies and segments the input into particular categories of motion. The resulting categorical information may be then used to describe the scene, extrapolate events, or as a part of a gesture recognition system. Applications beyond pen-based input are discussed. This paper contributes to pen based motion recognition research in two ways. First, a classification is performed based on a continuous sequence of observations, rather then feature extraction. Secondly, pen motion is transformed into a translation and rotation invariant representation prior to classification.
**Keywords:** Motion Recognition, Hidden Markov Models, Percepts, Gesture Recognition

## 1 Introduction

This project combines concepts from both handwriting theory and computational vision to create a system to classify the motion of a pen when writing/drawing. We have created a representation that divides the space of all possible motions in two-dimensions. To demonstrate one instantiation of the representation, we have implemented a system that classifies the motion of a pen on a drawing surface. This paper begins in section two which presents the ontology itself. Section three describes the motion recognition system. Section four summarizes the results of the system, and finally section five discusses future applications of both the ontology and the recognition system.

## 2 Motion Categories

The goal of the paper is to devise a system that captures useful properties of motion that occur, regardless of position or orientation. In other words the categories of motions we are interested in describing are rotation and translation invariant. Therefore, we do not attempt to describe motion in terms of position, but rather in terms of velocity and direction. To simplify this analysis the overall representation has been separated into two class hierarchies, one representing all possible angles and one representing all possible velocities. These categories, have been organized into a lattice /citejep1, ric1. The lattice is structured such that the most general cases are at the root of the lattice (pictured near the top in figures /refafig and /refvfig) and more specific cases children of the more general cases. Multiple inheritance is allowed. The general theory is the most specific state that is consistent with the input, is the most likely state. The angle lattice has 8 states, and the velocity lattice has 12 states, rendering a total of 96 states, although some states are extremely improbable.

### 2.1 Angle Structures

Figure 1 shows the angle lattice. Each state follows the general form of a curve, followed by a point of zero curvature, followed by a curve. The point of zero curvature (which is an inflection point) is indicated by a dot on the curve.

### 2.2 Velocity Structures

Figure 2 shows the velocity lattice. The Velocity lattice follows the same format as the angle lattice, where there is a type of velocity, followed by an inflection point in the direction, followed by another velocity categorization. Velocity can be described as

accelerating (A), decelerating (D), constant(C) or a subset of constant velocity, zero velocity (R). (E) represents a velocity event, which is either zero velocity (R) or undefined velocity (as in very rapid acceleration).

Figure 1: Angle Lattice

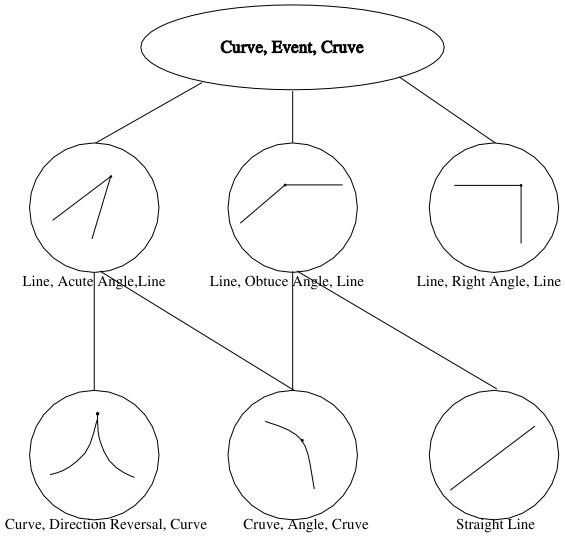
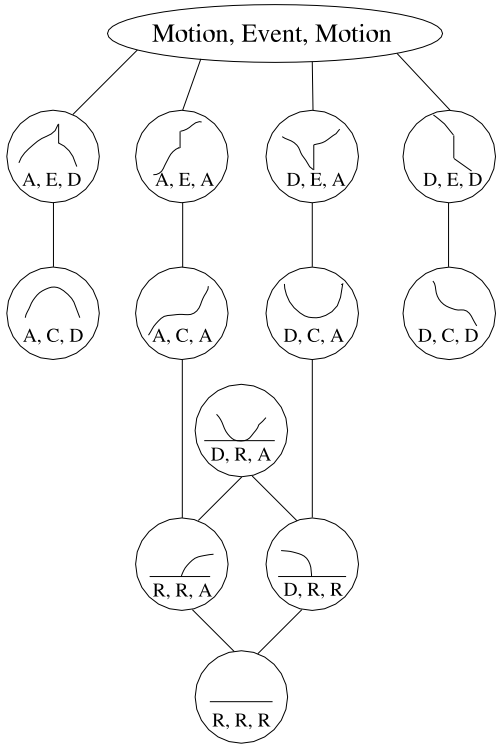



Figure 2: Velocity Lattice




# 3 Motion Recognition System

## 3.1 Overview

The recognition system consists of four components; data acquisition, pre-processing of the data, recognition using hidden Markov models, and finally segmentation. The input to the system is a stream of pen stroke information and the output is a set of tokens representing the motion categories. The output stream both provides a description of the movement and associated parameterization, and could be used as input to a gesture recognition system, or a event classifier. The goal of the system is to provide mid-level vision processing to facilitate a higher-level recognizer. The recognition system itself is based on a hidden Markov model (HMM) representation, described in section 3.2.

In order to simplify the recognition process, nine of the most common states from the above categories were selected to be recognized by the system. A graphical depiction of the states is shown in figure 8 and described in table 1. Each was manually segmented, and the recognizer was trained on it. The recognition system was then evaluated on a separate set of segmented data, and then on more complex non-segmented shapes that represented a number of motions, from which a stream of output categories was extracted.

## 3.2 Hidden Markov Models

Hidden Markov models, although existing for over three decades, first became popular in speech recognition systems in the late 1980's [4]. This section will only provide a brief overview of hidden Markov Models, to give the reader an intuition as to how the recognition system works. For a detailed description of HMM's, refer to [5, 4, 1]. For a different approach to pen based gesture recognition using HMMs, refer to [6]. This description of hidden Markov models is mainly derived from [4].

Hidden Markov models are based on the Markov processes, and are an extension of Markov chains. A Markov chain is a graph with nodes and arcs. Each node represents a state. Each arc represents a transition between nodes, and has an associated probability, which represents the probability of traveling across that arc from one node to another. A first-order Markov chain consists of a set of nodes and transition probabilities between nodes, and criteria based on observations on when to traverse an arc.

Then, given some input, it is possible to determine how well the Markov chain models the data. For instance, if the current state of the Markov chain is in node A, and based on the next observation the next state would travel along an arc to node B, if the probability of that arc is high, then the model fits the observations well. Hidden Markov Models, unlike Markov Chains have a second set of probabilities; the probability that we are in state A given the current observation. In other words, an observation may be probable in more the one state, and therefore the true state of the model is hidden. Finally, there are two basic types of Hidden Markov Models, discrete and continuos HMMs. Discrete HMMs have a fixed alphabet of observation symbols, where as continuos HMMs have a continuos range of inputs.

The components of a HMM therefore are: $O$, a sequence of observations. $N$, the number of states or nodes. $M$, size of the alphabet if this is a discrete HMM, or the number of mixtures per node if it is a continuous HMM, a state transition matrix $A = \{a_{ij}\}$, a probability distribution of the observation occurring in a certain state, $B$, and finally, an initial state distribution $\pi = \{\pi_i\}$ [4]. For continuous distribution HMMs there are a few more parameters. $B$, the probability distribution is replaced with a probability function:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} G(O, \mu_{jm}, \Sigma_{jm}) \qquad 1 \leq j \leq N \tag{1}$$

Where, $O$ is the current observation, G is a point distribution function, usually a Gaussian with mean $\mu$ and covariance matrix $\Sigma$. There can be more then one mixture per state, in which case $c_{jm}$ is the mixture coefficient of mixture $m$ in state $j$. All these parameters form the basis of the model of a HMM, $\lambda$.

To summarize all of the above, a HMM is initialized with a probability distribution $\pi = \{\pi_i\}$, which represents the probably of starting in state i. Using the translation matrix $A = \{a_{ij}\}$, we can then determine the probability of making a transition from state i to state j over all states. Finally, the function $b_j(O)$, allows us to determine the probability that the next observation $o_k$ was generated by state j. Combining all of these probabilities we can determine the overall probability that a sequence of observations O was generated by a particular HMM, $\lambda$, that is:

$$P(O|\lambda) \tag{2}$$

Therefore given a sequence of observations, $O$ and a number of different HMMs, $\Lambda = \{\lambda_1 \cdots \lambda_K\}$, we need to determine,

$$P(O|\lambda_i) > P(O|\lambda_j) \qquad \forall j, j \neq i \tag{3}$$

that is the model $\lambda_i$ which best explains the observations 0 out of all models $\Lambda$.

Calculating the probability $P(O|\lambda)$ as described above would be computationally prohibitive. Fortunately a more efficient procedure exists, refered to as the forward-backward algorithm [4], which stores intermediate variables using dynamic programming.

HMMs can be trained using an unsupervised learning algorithm called the Baum-Welch reestimation procedure [4], which can be thought of as an implementation of the EM-algorithm [1]. In the case of continuos HMMs the algorithm iterates, each time improving on the estimates of the parameters until a local maximum is reached, with the goal being to maximize $P(O|\lambda)$ for a number of training sequences O. Various modifications and alternative training procedures exist [1, 8].

Finally, it is worthwhile mentioning that there are countless variations of HMMs [1, 8, 4]. Most importantly, the structure and connectivity of HMMs can vary. The above discussion has focused on fully connected HMMs where there is a transition probability greater then zero between each node. Other types of HMMs include left-right HMMs, where there are only arcs between nodes $i$ and $j$ if $j \geq i$.

### 3.3 Data Acquisition

The data for the recognition system was acquired using a Wacom pen tablet. The pen's position on the tablet was sampled at a rate of 100Hz, with a resolution of 40 points per mm, with an accuracy of 0.5 mm. This provided very high-resolution data from which we could accurately extract velocity and angular measurements. Eighty observations of each state shown in figure 8 were taken. Forty were used in the training process, and the additional forty were used to validate the recognition system. Furthermore, 10 observations of each of the more complex shapes shown in figure 9 were recorded. It is possible that other input sources could be used such optical flow extracted from a video sequence.

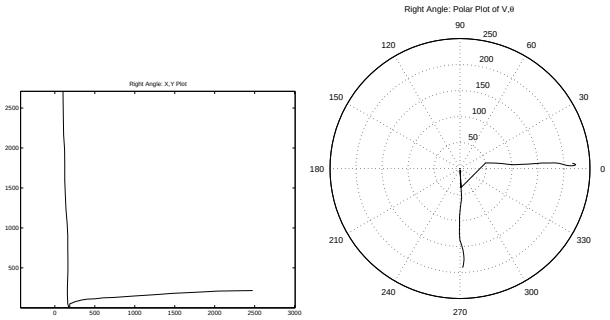
### 3.4 Pre-Processing

The raw data from the tablet includes the $X$ and $Y$ position of the pen tip, the pressure of the pen tip, and information about the tilt of the pen. Only the position data is used. The inputs to the HMM recognition system include the velocity, the current direction of the pen, and the direction change of the

pen relative to the start of the frame (more on what a frame is later), all of which are derived from the position.

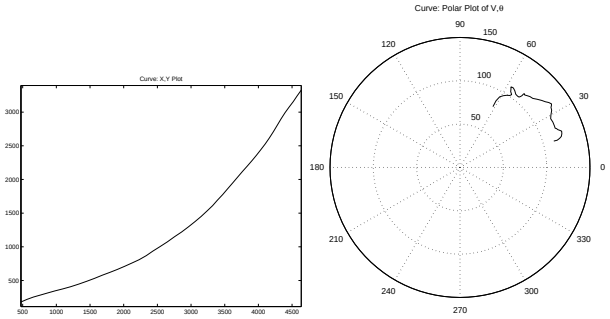
In other words, the HMM is never aware of the absolute position of the observations, nor explicitly the spatial relationships as we perceive them. Rather the inputs are the velocity of the observation and the angle of the velocity, which can be expressed graphically in polar form, as illustrated in figure 3 and 4. Typically there is a strong correlation between angle and speed. In the case of the right angle for example, the velocity typically slows almost to 0 as the angle changes, where as in the gentle curve, the velocity remains almost constant while the angle changes.

Figure 3: X,Y and Polar forms of a Right Angle




**Left:** X-Y Position plot of a right angle pen motion. **Right:** The same pen motion represented in polar form, indicating that angle of motion and the speed.

Figure 4: X,Y and Polar forms of a Curve




**Left:** X-Y Position plot of a curving pen motion. **Right:** The same pen motion represented in polar form, indicating that angle of motion and the speed.

The velocity is calculated separately on each axis from the position using the four-point central-difference approximation [7]. For example to calculate the velocity along the x-axis:

$$v_x = \frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12} \qquad (4)$$

The velocity is then smoothed using a median filter with a width of five samples. The purpose of this filter is to suppress any jittering that occurs as a result of the 0.5 mm error margin of the digitizing tablet.

The overall velocity can then be calculated,

$$v = \sqrt{v_x^2 + v_y^2} \qquad (5)$$

and the angle $\theta$ can be derived from the separate velocity vectors,

$$\theta = \arctan(\frac{v_x}{v_y}) \qquad (6)$$

from this, the change in $\theta$, $\theta'$ is calculated using a technique similar to (4), but compensating for the circular nature of angles.

Finally, the relative change in $\theta$ from the starting position is calculated. This is, if $\theta_{1...K}$ is a vector of changing angles, the relative change $\theta^*$ is

$$\theta_i^* = \theta_i - \theta_0 \qquad 1 \leq i \leq K \qquad (7)$$

The vectors $v$, $\theta'$, and $\theta^*$ become the observations for the HMM classification system described in section 3.5. This contrasts with the method in [6] where observations were based on $x$, $y$, $v_x$ and $v_y$. We chose to use $v$, $\theta'$, and $\theta^*$ because of they are rotation invariant, an important feature in gesture recognition systems and event classification systems, as a gesture or event can take place in any orientation yet have the same semantic meaning. This is similar to the work of [2], who have extracted speed and angle information from video sequences, to aid in video analysis and retrieval, however similarity between curves was the goal of their system, rather then classification. As shown below, experimentation seems to indicate that these vectors provide adequate differentiation between our classes for accurate classification.

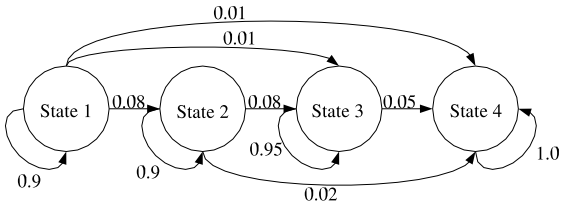
The final pre-processing step involves framing the data. For short pre-segmented observation vectors, as used in training and validating the HMM, the data was centered on the angle event and truncated to an equal length of 30 samples. For the longer sequence classification and segmentation, the data was broken up in to several frames of 30 samples each, each frame incremented by one sample from the previous frame. In this manner we are able to slide an 'observation window' over the data sequence.

Figures 6 and 7 shows plots of $v$, $\theta'$ and $\theta^*$ with respect to sample number (time) for a gentle curve and a right angle.

### 3.5 HMM Implementation

The HMM is the core of the motion recognition system. The HMM used in the recognition system is a continuous HMM with four nodes, and one mixture per node. Each mixture uses a Gaussian distribution to model error from the mean in velocity and angle. The structure of the HMM is a left to right HMM, implying that transitions back to nodes previously visited is not permitted. Figure 5 illustrates a sample HMM. Intuitively, one can think of each node having a mean that exemplifies the pen at a particular moment during the pen motion, and for the model to have a high probability of fitting the observations, it must traverse through the nodes without cycles.

Figure 5: Example left to right HMM



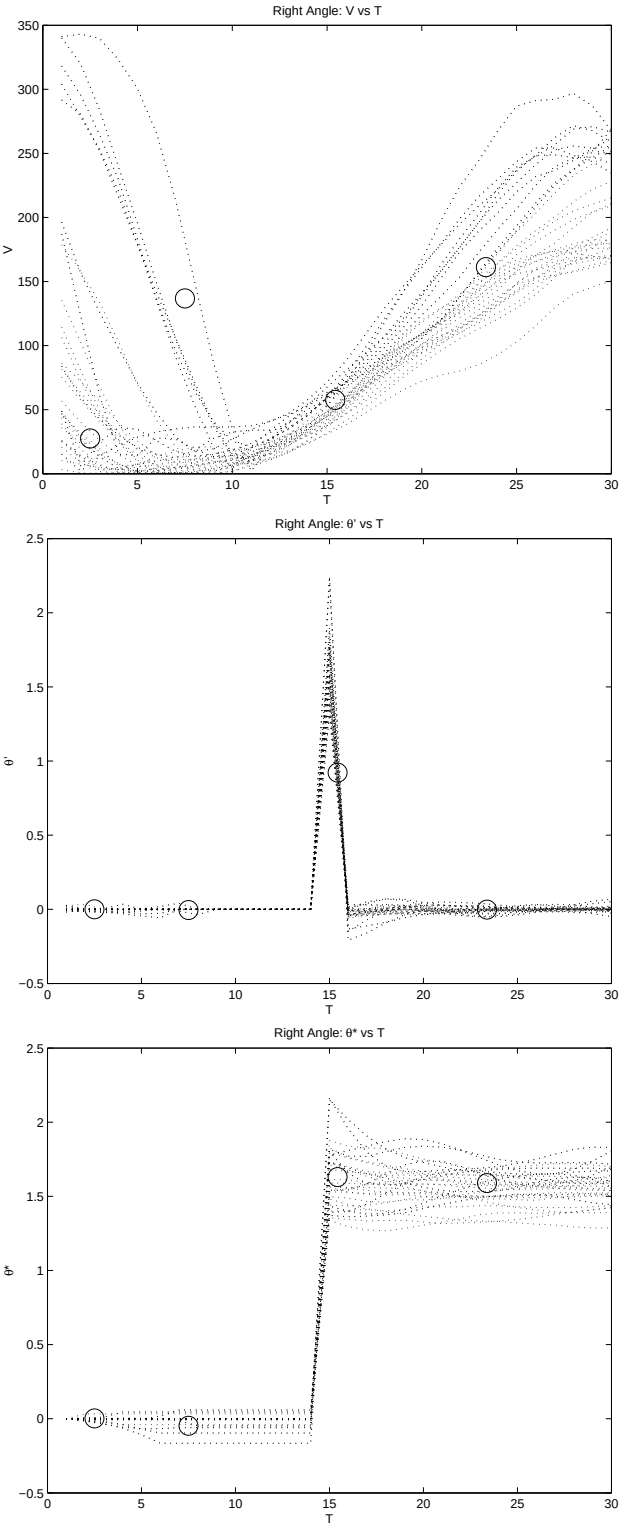

The above HMM shows the each state, and the transitions probabilities between states.

The HMM was trained using the standard Baum-Welsh [4] algorithm, modified to fix one node at the mean in the middle of the frame, so that we can be assured that spikes in $\theta'$ can be well represented. The training algorithm was run on forty samples of presegmented gestures for a maximum of 25 iterations. Occasionally the HMM found an unsatisfactory local maximum, in which case it was reinitialized with a new set of randomly generated priors and retrained.

Nine different HMM's were trained to recognize the nine different classes of motion illustrated in figure 8. Once training was completed, classification was performed as described in section 3.2 and equation (3). One practical difference, due to numerical precision issues relating to underflow from repeated multiplication of small probabilities, is that we represent $P(O|\lambda)$ in terms of log likelihood [4]. All code was implemented in Matlab.

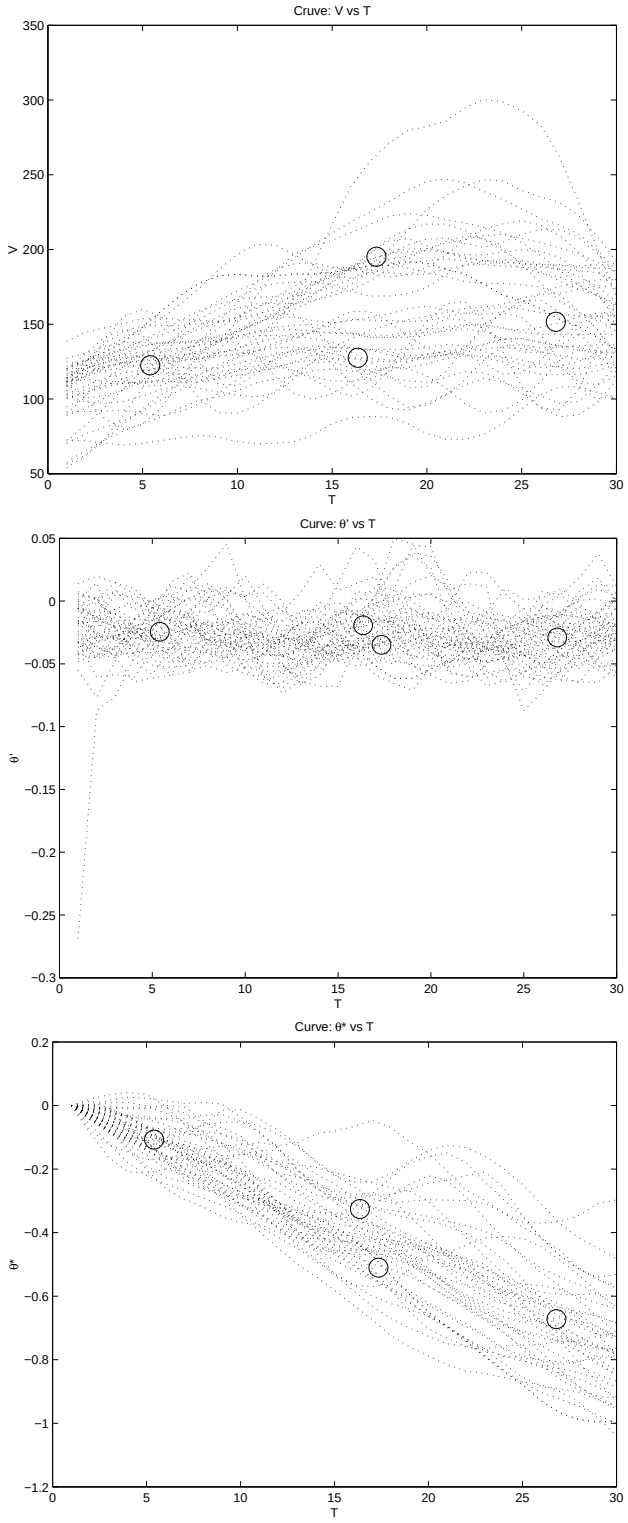
To illustrate this process, examine figures 6 and 7. The dashed lines represent each of the training observations, the ' ◯ 's represent the means for each of the nodes in the HMM.

Figure 6: Inputs to HMM for a Right Angle




See figure 3 for an example of the actual motion. **Top:** Speed with respect to time for right angle pen motions. Each pen motion used in training is shown as a dotted line. The means for each state of the HMM are shown as ◯. **Middle:** The change in direction with respect to time. **Bottom:** The direction of the motion, from the start of the motion, with respect to time.

Figure 7: Inputs to HMM for a Curve




See figure 4 for an example of the actual motion. **Top:** Speed with respect to time for curving pen motions. Each pen motion used in training is shown as a dotted line. The means for each state of the HMM are shown as ◯. **Middle:** The change in direction with respect to time. **Bottom:** The direction of the motion, from the start of the motion, with respect to time.

### 3.6 Segmentation

The sliding window approach described in section 3.4 produces a vector of most probable classifications. Segmentation comes naturally to this vector which typically involves large regions of similarly classified motion. Initially the stream is simply segmented into similarly classified adjacent motions, and then segments below a certain threshold (typically five samples) are discarded.

## 4 Results

The initial testing on the recognition system consisted of two experiments. The first experiment classified pre-segmented data. The second experiment consisted of using a sliding window to classify and segment a larger more complex motion. The results of both experiments are shown below.

The final output of the system consists of the segmented symbols, the parameters associated with the symbols (so angle and distance can be extracted, for each segment), as well as probabilities for each of the other classes for that segment.

### 4.1 Experiment One: Pre-segmented Data

In the first experiment, the forty pre-segmented observations of each class used for training and the forty pre-segmented observations of each class, collected at the same time as the training samples but not represented in the training samples were processed by the recognition system. Both the training data and the testing data were composed at a number of different angles. The results of this experiment are summarized in table 2. The classes are illustrated in figure 8, and the meaning is summarized in table 1.

Table 1: Semantic Meaning of the Classes

| Class | Interoperation |
|---|---|
| 1 | Right Angle CCW |
| 2 | Right Angle CW |
| 3 | Acute Angle CCW |
| 4 | Acute Angle CW |
| 5 | Obtuse Angle CCW |
| 6 | Obtuse Angle CW |
| 7 | Curve CCW |
| 8 | Curve CW |
| 9 | Straight Line |

Where CW refers to movement in a clockwise direction and CCW refers to movement in a counter clockwise direction.

Figure 8: Nine Primary States

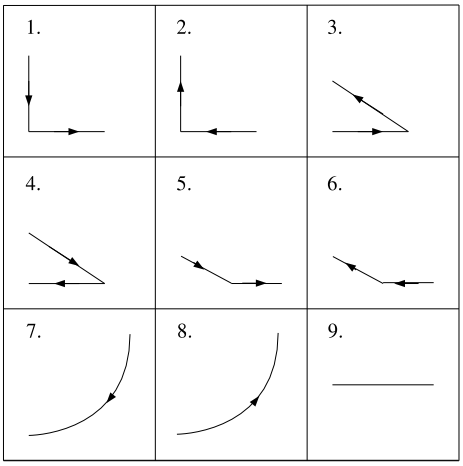



Table 2: Accuracy of Pre-segmented Data

| Class | Training Data | Testing Data |
|---|---|---|
| 1 | 100 % | 100 % |
| 2 | 97.5 % | 100 % |
| 3 | 100 % | 100 % |
| 4 | 100 % | 100 % |
| 5 | 100 % | 100 % |
| 6 | 97.5 % | 100 % |
| 7 | 92.5 % | 87.5 % |
| 8 | 97.5 % | 100 % |
| 9 | 100 % | 100 % |

Overall, the HMM recognition system proved to be very accurate on pre-segmented data. It is curious to note in the cases of classes 2,6 and 8 the HMM preformed better on the testing data then the training data, although in each case it simply indicates that it miss-classified one sample.

## 4.2 Experiment Two: Complex Motion Data

Figure 9 contains illustrations of the more complex movements classified by the system. To test the accuracy of the system, the resulting classification was manually analyzed to determine if the classification seemed reasonable. For instance, while a human may perceive a straight line, it is reasonable for the computer to see a series of obtuse angles or very gentle curves. Table 3 shows an example output for each of the examples. Segments in *italics* are not consistent with our perception of the motion. The meaning of each class is summarized in table 1.

Figure 9: Complex Motion Examples

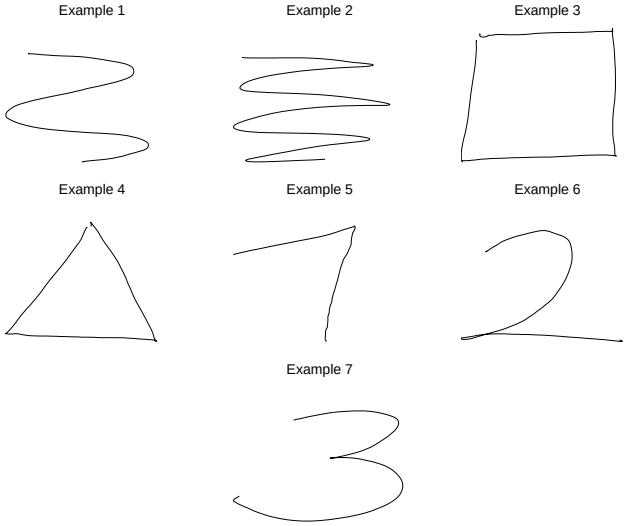



Table 3: Accuracy of Complex Motion

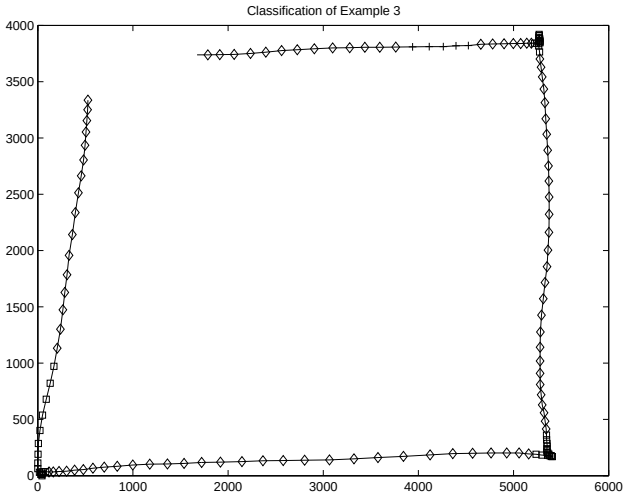
| Example | Segmentation Results |
|---|---|
| 1 | 6,*2*,7,8,*3*,8,6 |
| 2 | 4,3,6,4,3,4 |
| 3 | 6,5,1,5,6,1,6,5,1,9 |
| 4 | 6,7,3,9,6,3,9 |
| 5 | 9,2 |
| 6 | 6,*2*,7,4,7 |
| 7 | 6,*4*,*2*,6,3,*4* |

The results from this experiment are not as conclusive as from experiment one. Clearly on some

examples, the system did very well. In example 3 the system located all three corners, and identified them as right angles, and in example 4 the system located the two corners, and identified them as acute angles. However, in example 7, the system was detecting right angles and acute angles where clearly there were none.

Figure 10 shows the classification of example 3.

Figure 10: Classification of Example 3




In this figure, □ indicates a classification as a right angle (Category 1), ◇ represents a obtuse angle (Category 5 or 6) and + indicates a straight line (Category 9).

# 5 Conclusions and Future Work

In conclusion, we have presented a representation of possible types of 2-D motion and have presented a system that can recognize and extract a subset based upon hidden Markov models.

The recognition system attempts to model the regular structure that underlies the motion, while at the same time maintaining invariance to rotation and translation. The results returned by the classifier for pre-segmented data are very accurate; the results returned on real motion sequences have promise. Normalizing the velocity data is expected to improve these results and is currently being investigated. Directly building on this work consists of establishing a preference lattice, to aid in the selection of the appropriate state, when multiple candidates are returned from the HMM, and an algorithm to optimally segment the results, based on dynamic programming similar to [3].

Other extensions being investigated consist of adding global parameters to the HMM, such as the starting state, and other learning algorithms, such as the parameterized HMM algorithm discussed in [8].

The outputs from this classification system have a wide range of possible applications. A gesture recognition system could use the output of the recognizer, which too is currently being explored.

# References

[1] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *Hidden Markov models: Applications in computer vision*, pages 9–42, 2001.

[2] James J. Little and Zhe Gu. Video retrieval by spatial and temporal structure of trajectories. *SPIE Storage and Retrieval for Media Databases*, 2001.

[3] R. Mann, A. D. Jepson, and T. El-Maraghi. Trajectory segmentation using dynamic programming. *Proc. Sixteenth International Conference on Pattern Recognition*, 2002.

[4] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, feb 1989.

[5] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Mag.*, pages 4–16, jan 1986.

[6] Donald O. Tanguay. Hidden markov models for gesture recognition. Master's thesis, Massachusetts Institute of Technology, 1995.

[7] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, Upper Saddle River, New Jersey, 1998.

[8] Andrew D. Wilson and Aaron F. Bobick. Hidden markov models for modeling and recognizing gesture under variation. *Hidden Markov models: Applications in computer vision*, pages 123–160, 2001.