Online, Interactive Learning of Gestures for Human/Robot Interfaces

Christopher Lee and Yangsheng Xu
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA

Abstract

We have developed a gesture recognition system, based on Hidden Markov Models, which can interactively recognize gestures and perform online learning of new gestures. In addition, it is able to update its model of a gesture iteratively with each example it recognizes. This system has demonstrated reliable recognition of 14 different gestures after only one or two examples of each. The system is currently interfaced to a Cyberglove for use in recognition of gestures from the sign language alphabet. The system is being implemented as part of an interactive interface for robot teleoperation and programming by example.

1 Introduction

If we are to fully harness the potential of robotic technology, we will have to move beyond simple keyboard/mouse/teach-pendant style robot programming and create comprehensive frameworks for productive realtime interaction between robots and humans. The motivations behind this kind of interaction include increasing the effectiveness of teleoperation, enabling people to interactively teach robots new tasks or refine their skills, and allowing people to more effectively control systems such as semi-autonomous airplanes or systems for automated monitoring of industrial plants. As people interact with machines which are autonomous and highly complex, they must be allowed to focus their attention on the content of their interaction rather than the mechanisms and protocol through which the interaction occurs. This is best accomplished by making the style of interaction more closely resemble that to which they are most accustomed: interaction with other people.

One of the first efforts towards realizing this new style of human/robot interaction is current research on programming by example. In this methodology, a robot learns to perform a task by observing a human teacher. Much effort is being directed toward the research of systems for gesture/observation-based programming of robot systems. Tung and Kak [1] demonstrate automatic learning of robot tasks through a DataGlove interface. Kang and Ikeuchi [2] developed a system for simple task learning by human demonstration. Voyles [3] developed a system for gesture-based programming via a multi-agent model.

One capability which is currently lacking in systems such as these is a mechanism for online teaching of gestures with symbolic meanings. Most gesture recognition systems either require some explicit programming, or in the case of neural-nets, require offline training of model parameters. Such systems are thus unsuitable for interactive applications where gestures must be taught and then recognized without waiting for an offline training or programming phase. A teach-by-demonstration system should also be able to learn a new gesture or skill in an online manner and with a very small number of examples. This simplifies the teaching process because it more closely resembles the manner of instruction which we commonly use to train people.

In this paper, we present a gesture recognition system which can interactively recognize gestures and learn new gestures online with as few as one or two examples. It is also able to update its model of a gesture iteratively with each example it recognizes.

2 Approach

Our goal is to make a system which can not only interact with a user by accurately recognizing gestures, but which can learn new gestures and update its understanding of gestures it already knows in an online, interactive manner. Our approach is automated generation and iterative training of a set of Hidden

Markov models which represent human gestures. Using this approach, we have built and tested a system which recognizes letters from the sign language alphabet using a Virtual Technologies 'Cyberglove'.

The most basic assumption we make about the nature of human gestures are that they are "doubly stochastic" processes. These are Markov processes whose internal state is not directly observable. For each state transition in such a process, the system generates an observable output signal whose value depends on a probability distribution which is fixed for each internal state. A model of such a system is called a Hidden Markov Model (HMM). Modelling human gestures as HMMs allows us to deal with the highly stochastic nature of human gesture performance.

For computational simplicity, we assume that the the HMMs are 'discrete' HMMs. These are HMMs whose observable outputs are members of a finite set of symbols. Before we can use discrete HMMs to model gestures, we must therefore preprocess the raw data from the gesture input device into a sequence of discrete symbols.

Gesture recognition is, to a certain extent, a process of data compression, where we input a large amount of raw data and output a signal value that represents the classification of the gesture. This compression process is composed of two stages. The greatest data-reduction occurs when the preprocessor converts the large, multichannel stream of data from the gesture input device to the sequence of discrete observable symbols. We reduce about 0.5-1.0 Kbytes of data from the Cyberglove to a sequence of 5-10 observable symbols from a set of 32 and 256 symbols, which is a reduction of about 100:1. Since the online portion of our gesture learning process is limited to the modification of the parameters of HMMs, this data-reduction greatly increases the speed and simplicity of the online learning process by focussing the HMM on modelling specific features of the signal.

3 Interactive training

Our project is to build a system which allows for interactive, online training. In our system, each kind of gesture is represented by an HMM, a list of example observation sequences, and an optional action to be performed upon recognition of the gesture. Our concept of interactive training is currently based on the following general procedure:

1. The user makes a series of gestures.
2. The system segments the stream of data from the input device into separate gestures, and in realtime, tries to classify each gesture.
   (a) If the system is certain about its classification of a gesture, it immediately performs an action associated with that gesture (if one has been specified). Such an action could be passing the result of the classification to a higher-level HMM, or sending a command to a robot.
   (b) If the system is in any way unsure about its classification of a gesture, it queries the user for confirmation of its classification. The user either:
      - confirms the system's classification, or
      - corrects the classification, or
      - adds a new gesture class to the system's bank of gesture models.
3. The system adds the symbols of the encoded gesture to the list of example sequences of the proper gesture model, then updates the parameters of that model by retraining the HMM on the accumulated example sequences.

We have found that in our implementation, recognition of the gesture and automatic update of the HMM through the Baum-Welch algorithm is fast enough not to be noticeable during normal use of the system. This provides the system with a truly interactive character. For example, a user controlling a robot through the gesture system could perform a gesture which the system has not seen before, and the system would immediately respond by asking what kind of gesture it is. The user could respond that it is a "halt" gesture, and that the robot should stop its current motion when that gesture is made. The next time the user performs that gesture, the system should recognize it and immediately halt the motion of the robot.

4 Recognition and Learning with HMMs

Hidden Markov Models [4] are commonly used for speech recognition, but have also been used for characterizing task information and human skills for transfer to robots in telerobotic applications [5, 6].

A Hidden Markov Model is a representation of a Markov process which cannot be directly observed (a "doubly stochastic" system). The discrete form of the HMM is represented by three matrices, $\lambda = (A, B, \pi)$. The matrix $A = a_{ij}$ specifies the probability that the internal state will change from i to j. $B = b_{jk}$ represents the probability that the system will generate the observable output symbol k on transition to state j. The third component of a Hidden Markov Model is a vector π indicating the distribution of probability that any given state is the initial state of the hidden Markov process.

There are three problems commonly associated with Hidden Markov models [4]: (1) determining the probability with which a given sequence of observable symbols would be generated by an HMM, (2) determining the most likely sequence of internal states in a given HMM which would have given rise to a given sequence of observable symbols, and (3) generating an HMM that best 'explains' a sequence or set of sequences of observables. We are directly concerned with the first problem for gesture recognition, and the third problem for generating the HMMs used in gesture recognition.

Gesture recognition. The problem of recognizing a gesture from a given set of input data is an example of problem 1. First, raw input data from the input device is preprocessed into a sequence of discrete observation symbols $O = O_1 O_2 O_3 \ldots$. Then it is determined which of a set of HMMs, each modeling a different gesture, is most likely to have generated that sequence: $L = \text{argmax}[P(O|\lambda_l)]$. In addition, the system determines if there is an ambiguity between two or more gestures (the probabilities of the most likely gestures are too close to one another) or if no known gesture is similar to the observed data (the probability of the most likely gesture is too small).

Learning gesture models Developing the HMM which will be associated with a gesture is an example of problem 3. The algorithm which is commonly used for this purpose is the Baum-Welch (BW) algorithm. Baum-Welch uses an iterative expectation/maximization process to find an HMM which is a local maximum in its likelihood to have generated a set of 'training' observation sequences. Although the space of possible HMMs for a given HMM structure is generally full of many of these local maxima, it has been observed that most work similarly well for practical use in modeling doubly stochastic systems.

While training of HMMs is normally a batch process, where many examples of a given gesture are used simultaneously as inputs to the Baum-Welch algorithm, we use a different approach. We begin with one or some small number of examples, run BW until it converges, then iteratively add more examples, updating the model with BW after each one. This allows for an online, interactive style of gesture training. Section 7 compares the results of this kind of incremental training to batch-style processing.

5 Signal preprocessing

Since we are using discrete HMMs, we need to represent gestures as sequences of discrete symbols. We must therefore preprocess the raw gesture data, which in our case are values of 20 joint-angles in the hand, estimated from 18 sensors in the Cyberglove at about 10 Hz.

The first choice we must make in performing this preprocessing is whether we want to generate a one-dimensional sequence of symbols or a multi-dimensional sequence. The multi-dimensional sequence may be used as input to a multi-dimensional HMM. If we assume that the dimensions of the observable sequence are dependent only on the internal state and not on one another, then the multi-dimensional HMM will have a single A matrix and multiple B matrices, one for each dimension of the output symbols. Although it makes some intuitive sense to train a multi-dimensional HMM which takes as inputs, say, 5 dimensions of symbols (one for each finger), we chose for simplicity to look at the hand as a single entity, and generated a single-dimensional sequence of symbols which represent features with respect to the entire hand. The specific preprocessing algorithm we chose for this purpose is inspired by work done with HMMs in the speech community—it is vector-quantization (VQ) of a series of short-time fast Fourier transforms (STFFTs) of the signal from the input device.

Figure 1 shows the flow of data through the preprocessor. 20 channels of joint-angle data from the hand are read from the Cyberglove [a]. This data is segmented into separate gestures [b], and resampled by cubic interpolation [c]. Each channel is then broken into a series of overlapping time-windows of 4-8 readings, and each time-window is smoothed with a Hamming function before being passed to the FFT routine [d]. The power spectra of the 20 channels are then concatenated [e] to form a large vector [f]. This vector represents the position and dynamic characteristics of the gesture during the time-window. Each of these large vectors is turned into an observable symbol by a vector-quantizer [g].

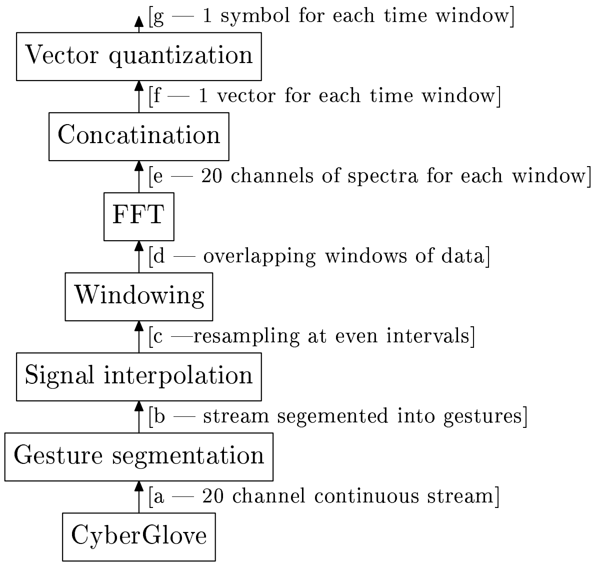


Figure 1: Data flow in the preprocessor

The vector quantizer encodes a vector a by returning the index K of a vector c_K in a set of vectors called the 'codebook' ($c_K \in C$) which is closest to a in the L_2-norm sense (i.e. $K = \mathrm{argmin}(c_k \cdot a)$). The codebook is a set of vectors which is believed to be representative of the domain of vectors to be encoded. We generate this codebook using the LBG algorithm [7] on a representative sample of gesture data. Codebook generation is an offline process.

Because the preprocessor is coded as a data filter, a symbol is sent to the gesture-recognition system as soon as enough data has been read from the glove to generate it. This allows the system to eventually be reconfigured to use HMMs for online segmentation of continuous gestures rather than performing segmentation as a separate step.

This preprocessor is not task specific. It assumes only that all dimensions of the data are related, that the gesture itself is a process which evolves fairly smoothly over time, that the interesting features of the gesture are amenable to clustering in the spectral domain and are consistent in nature over time. It may thus be used without modification for recognizing gestures such as handwriting, facial expressions, or dance motions as well as hand gestures. Note that specially developed, task-specific signal-preprocessors could be expected to perform better for recognition of specific kinds of gestures by leveraging expert *a priori* knowledge of the structure of the gestures, and would be necessary for signals which do not satisfy the assumptions stated above.

6 Implementation

Our gesture segmentation procedure is very simple, relying on the hand of the operator to be still for a short time between gestures. Another possible tool for segmentation is an acceleration threshold. This is useful for segmentation when the hand does not stop between gestures, and can be combined with the velocity-based segmentation.

Figure 2 shows the organization of our system. A Cyberglove is interfaced to a Sun Sparcstation via an RS-232 serial connection, and sends readings of joint angles in the user's hand to a data-collection program at about 10 Hz. Each reading that is part of a gesture is marked with a timestamp, and sent to the gesture-recognition program via either a TCP-based UNIX socket or a temporary file. The socket connection is used for normal operation, and the temporary file is used for creating databases of gesture data for offline generation of vector-codebooks, and for automated testing and tuning of system parameters.

After the data for a gesture is preprocessed by the algorithm specified in Section 5, it is evaluated by all HMMs for the recognition process, and then used to update the parameters of the proper HMM. For recognition of hand gestures, we used 5-state Bakis HMMs. A Bakis HMM is one where the system is restricted to only move from a given state to the same state or one of the next 2 states. A 5-state Bakis HMM may move from state 1 to states 1, 2, or 3, and from state 4 to state 4 or 5 (there is no state 6). This restriction encodes the assumption that the gestures we are classifying are in general a simple sequence of motions, and non-cyclical in nature. Using this HMM structure, we performed tests to determine the ability of the system to accurately recognize gestures and to optimize the parameters of the preprocessor.

The data-collection program for the Cyberglove is written in C, and the interactive HMM recognition/training/GUI program is written in a combination of C, Tk, and a language closely related to Scheme–Guile. Although matrix operations are coded in C for speed, the outer loops of Baum-Welch, gesture-recognition, and all other code is currently executed in interpreted Scheme. Even with the majority of the code written in Scheme, gesture-recognition and update of the proper HMM occurs in a fraction of a second on a Sparcstation.

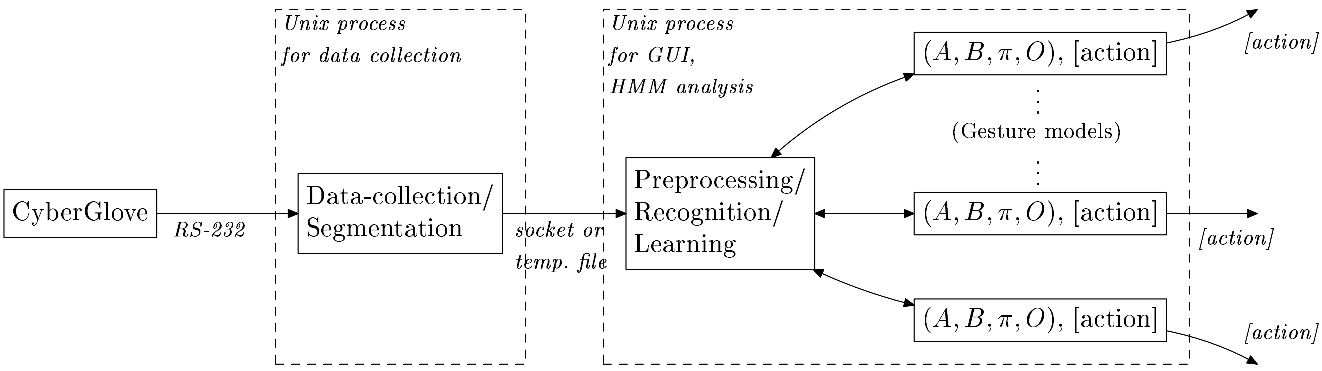


Figure 2: Data flow for learning system

7 Confidence measure

We defined a simple evaluation function to examine the classification power of our algorithm. The function indicates misclassifications and their severity, as well as the system's confidence in its correct classifications.

Our evaluation function is

$$V = \log_{10}\left(\sum_i P_{E_i}/P_C\right),$$

where P_C is the probability that the observation sequence would be created by the correct gesture model, and P_{E_i} is the probability that the gesture symbols would be created by the ith incorrect model. Therefore, if $V < 0$, we have correct classification, and if $V > 0$, the gesture is incorrectly classified. If $-1 < V < 1$, the classifier should indicate that its classification is suspect. If $V < -2$, the system has made the correct classification and is very confident of the classification.

We plotted the performance of the system using the following procedure:

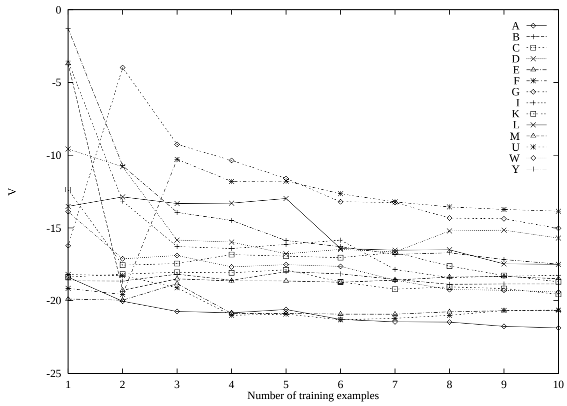
1. Train each gesture model with one example.
2. Test the classification of 20 test examples of each kind of gesture (the models are never trained on these examples).
3. Plot the average value of the V verses the number of examples of gestures each model has seen.

This procedure allows us to judge the performance of the algorithm with respect to learning rate and overall confidence. It also allows us to test the effect of varying parameters such as the number of vectors in the codebook (the number of observable symbols), the size of the STFFT windows, and the resampling timestep for the signal interpolator.

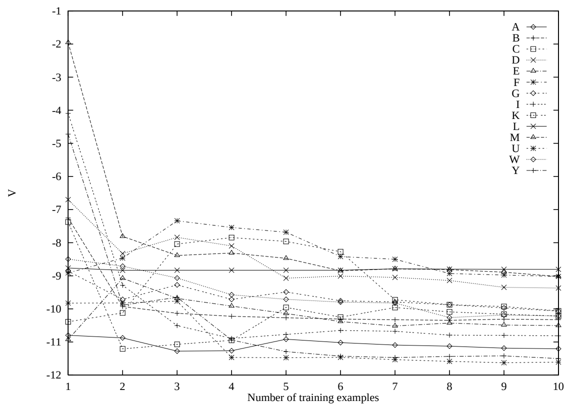
For our test, we picked 14 letters from the sign language alphabet which were amenable to VQ clustering and unambiguous with respect to general hand orientation, since we did not using the Polhemus 6D position/orientation sensor for the hand. The letters we used are A, B, C, D, E, F, G, I, K, L, M, U, W, and Y. The final positions for two of these gestures are shown in Figure 4.

Two plots from our tests are shown in Figure 3. They demonstrate that the system is very reliable for both these sets of preprocessing parameters. The average value of V in all cases is less than zero, indicating reliable classification. Measuring percentage of classification errors, the trail of plot (b) had 1% error after 2 examples and less than 0.1% error after 4 examples. The trial of plot (a) had 2.4% error after two training examples, and made no classification errors after seeing 6 examples.

The iterative training of the HMMs usually results in models which are close to the quality of batch-trained HMMs when we compare the likelihood that the training data would be generated by the models. In a few cases, however, batch training did much better. This is probably the result of early training examples biasing an HMM toward a poor local maximum. Fortunately, batch training from a random HMM is a rapid process, and can be easily used to try to improve HMM models during a couple seconds while the system is not doing realtime recognition. Our results show that this is not generally necessary, however, as iteratively trained HMMs work well enough in classifing gestures.



(a) 128 symbols, 10 Hz resampling, 4 point data windows



(b) 64 symbols, 20 Hz resampling, 8 point data windows

Figure 3: Evaluation of classification process

Figure 4: Final positions of two gestures: A and C

8 Conclusion and Discussion

We have demonstrated an efficient and reliable system for online learning and recognition of gestures. Online learning of gestures in an interactive system can be used to make cooperation between robots and humans easier in applications such as teleoperation and programming by demonstration.

The number of gestures we can classify this accurately is currently limited by the number of observable symbols the preprocessor generates. Thus at this time, systems which perform offline training now can recognize a larger gesture vocabulary. Fels and Hinton [8], for example, use offline-trained neural networks to recognize 66 root words, each with up to 6 endings. We are working on increasing the potential size of the gesture vocabulary of our online system by having the preprocessor generate 5 dimensions of symbols (one for each finger) as input to a multidimensional HMM. We are also working on a mode where we can turn off learning of new gestures and the gesture-segmenter, and have the HMMs automatically segment continuous gestures. Finally, we are investigating the use of this system for human to robot skill transfer and teleoperation.

References

[1] C. P. Tung and A. C. Kak, "Automatic learning of assembly tasks using a dataglove system," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, pp. 1–8, 1995.

[2] S. B. Kang and K. Ikeuchi, "Robot task programming by human demonstration," in *Proceedings of the Image Understanding Workshop*, 1994.

[3] R. Voyles, "Tactile gestures for human/robot interaction," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1995.

[4] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, pp. 4–16, January 1996.

[5] J. Yang, Y. Xu, and C. Chen, "Hidden markov model approach to skill learning and its application in telerobotics," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 5, pp. 621–631, 1994.

[6] Y. Xu and J. Yang, "Towards human-robot coordination: skill modeling and transferring via hidden markov model," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1906–1911, 1995.

[7] A. Gersho, "On the structure of vector quantizers," *IEEE Transactions on Information Theory*, vol. IT-28, no. 2, pp. 157–166, 1982.

[8] S. S. Fels and G. E. Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," *IEEE Transations on Neural Networks*, vol. 4, pp. 2–8, January 1994.