CHAPTER 14

Understanding of User Behavior in Immersive Environments

Cyrus Shahabi, Leila Kaghazian, Soham Mehta, Amol Ghoting, Gautam Shanbhag, and Margaret L. McLaughlin

Immersive evironments can facilitate the virtual interaction between people, objects, places, and databases. Immersion has several varied practical applications. It can serve as an aid to engineering applications. Immersion can also be used to understand and aid the disabled. These environments result in the production of large amounts of data for transmission and storage. Data types such as images, audio, video, and text are an integral part of immersive environments and many researchers in the past have addressed their management. However, we have identified a set of less familiar data types, collectively termed *immersidata* (Shahabi, Barish, Ellenberger, Jiang, Kolahdouzan, Nam, & Zimmermann, 1999) that are specific to immersive environments. Immersidata are produced as a result of the user's interactions with an immersive environment.

Haptic data is a kind of immersidata that is used to describe the movement, rotation, and force associated with user-directed objects in an immersive environment. We use the CyberGlove as a haptic user interface to an immersive environment. The CyberGlove consists of several sensory devices that generate data at a continuous rate. The acquired data can be stored, queried, and analyzed for several applications.

In this chapter, we focus our attention on the analysis of haptic data with the objective of modeling these data in a database. A large number of diverse applications use haptic data. Each such application may need haptic data stored and modeled at different levels of abstraction. For now, we consider three levels of abstraction. First, in Shahabi, Barish, Kolahdouzan, Yao, Zimmermann, and Zhang (2001), we made our first attempt to model haptic data at the lowest level of abstraction. There, we dealt with raw haptic data conceptualized as time-series data sets. Such a modeling approach can be used for training applications such as comparing a teacher's and a student's session with the CyberGlove, to measure the student's proficiency at following the teacher. Second, in this chapter we move a level up from our previous work in using raw haptic data by trying to understand the *semantics* of hand actions, and we employ several learning techniques to develop this understanding. The application that we focus on is *limited vocabulary American Sign Language recognition* that involves the translation of American Sign Language (ASL) to spoken words. Finally, for the third level of abstraction, there exists a class of applications that need to analyze *preprocessed* data, as opposed to analyzing raw haptic data. An example would be the application of detecting the grasping behavior of the hand. This application might need the speed of the hand in space at a certain instant of time. We intend to study this final level of abstraction as part of our future work.

We analyzed the raw haptic data acquired from the CyberGlove to recognize different hand signs automatically. We investigated three different analysis techniques and evaluated their accuracy over a 10-sign vocabulary. First, Decision Tree was used for the supervised classification of haptic data. This technique generates decision trees derived from a particular data set. In particular, we used C4.5 Decision Tree to classify haptic data into static signs. Our experiments show that this technique can classify haptic data with an average error of 22%. Second, we utilized Bayesian Classifier for the classification of haptic data. Bayesian Classifier is a fast-supervised classification technique that generates fine-grained probability estimates over the data set. In our experiments, this technique resulted in an average error of 15.34%. Bayesian Classifier appears to be the fastest classification technique providing the best classification accuracy for our experiments. Finally, we used Supervised Neural Networks as a classification technique for the recognition of both static and dynamic signs. We show that with Neural Networks, static signs can be recognized with an average error of 20.18%.

Our research is distinct and novel in the following three respects. To begin with, we are distinct with respect to the framework we have used for our research and experiments. The framework is based upon the environment provided by the CyberGlove from Immersion Corporation. All our analysis and experiments were performed on raw haptic data without any kind of preprocessing. The analyzed data sets were collected and recorded by us, using an application developed at our laboratory. In addition to a novel framework, we have taken a new approach to modeling haptic data, which is based upon various learning techniques. To the best of our knowledge, we are the first to use Decision Trees for the analysis of raw haptic data. Bayesian Classifier was used in the past for the analysis of preprocessed haptic data; however, as far as we know this classification technique was never used for the analysis of raw haptic data. Neural Networks have been used for the analysis of haptic data in many research efforts in the past. Our contributions to the analysis of haptic data using Neural Networks include the use of Back-Propagation Neural Networks for static sign recognition and the use of Time-Delay Neural Networks for dynamic sign recognition. The comparison of these three techniques within the same environment and experimental setup is also novel and unique. Finally, the ultimate objective of our research is to model and store haptic data at different levels of abstraction. Consequently, each kind of application can use haptic data stored at the level of abstraction that is most suited to its analytical needs.

The remainder of this chapter is organized as follows. First, we describe how we acquired haptic data using the CyberGlove. Our proposed techniques, fixed sampling, group sampling, and adaptive sampling, allow us to also take the time dimension into consideration, which can be used for the analysis of haptic data for dynamic sign recognition. Next we explain the three learning techniques that we used for sign recognition. The results of our experiments in comparing the three analysis techniques are then reported. Finally, we cover other research efforts in sign language recognition and outline our future research plans.

DATA ACQUISITION

The development of haptic devices is in its infancy. We have focused our research and experiments on the CyberGrasp exoskeletal interface and accompanying CyberGlove, which consists of 33 sensors (Table 14-1). We use the CyberGrasp SDK to write handlers to record sensor data for our experiments whenever a sampling interrupt occurs.

The rate at which these handlers are called is thus the maximum rate at which we can sample the input signal, and it is a function of the CPU speed. We developed a multi-threaded double buffering technique to sample and record data asynchronously. One thread is associated with responding to the handler call and copying sensor data into a region of system memory. A second thread asynchronously writes this data to disk. The CPU was never 100% utilized during this process. This prevents our recording process from interfering with the rendering process. There is obvious room for optimization here, as we can run our experiments on a dual processor machine and adjust the priority for the second thread. We used 10 letters (A to I and L) from ASL for our experiment. We term each of these 10 letters a *sign*. The 22 sensor values (excluding sensors 23 to 33 in Table 14-1) are recorded in a log file for each sign made by a *subject,* termed as a *session*. Each session log file contains thousands of rows of sensor values sampled at some frequency, which depends on the sampling technique used. We denote each such row as a *snapshot*. We thus have thousands of snapshots for each session.

Table 14-1: CyberGrasp sensors.

Sensor Number	Sensor Description	Sensor Number	Sensor Description
1	Thumb roll sensor	15	Ring middle abduction
2	Thumb inner joint	16, 17, 18	Pinky inner, middle, outer joint
3	Thumb outer joint	19	Pinky ring abduction
4	Thumb index abduction	20	Palm arch
5, 6, 7	Index inner, middle, outer joint	21	Wrist flexion
8, 9, 10	Middle inner, middle, outer joint	22	Wrist abduction
11	Middle index abduction	23, 24, 25	*x*, *y*, *z* location
12, 13, 14	Ring inner, middle, outer joint	26, 27, 28	*x*, *y*, *z* abduction
		29 to 33	Forces for each finger

Sampling Techniques

To record several snapshots for each static sign made within a session, we need to sample the values of sensors for each subject making a sign. Moreover, ASL is not restricted to just static signs. It has some dynamism in signs (e.g., the letter ‘J’ involves moving the hand) and in words (e.g., ‘BOX’ is represented by depicting a rectangular shape). Hence, the time dimension needs to be considered while recording the data. Thus for both static and dynamic signs, the time at which each sensor is sampled impacts the storage and the exact representation of the data. If a sensor value is recorded too frequently, then we will obviously get a very accurate representation of the sensor data, but the storage requirements and the transmission requirements increase. On the other hand, if the sensor value is recorded intermittently, we would save storage space but at the same time, we would run the risk of recording an inaccurate representation of the data. Thus sampling the sensors at the rate that would lead to lower storage space requirements and better accuracy is central to the task of data acquisition for any haptic device. We designed and implemented the following sampling techniques for our experiments (see Shahabi et al., 2001, for details):

1. **Fixed Sampling**: Fixed sampling can be approached in two ways. One approach is to use the maximum sampling rate *rmax* allowed by the Software Development Kit. While this technique is easy to implement, it is wasteful since it records data for each sensor at each possible opportunity regardless of the sensor type or the semantics of the session. A more efficient method in-

volves finding the minimum sampling rate *r0* required for the entire sensor set and then using that as the sampling rate. The disadvantage to this approach is that we need to identify *r0* before we start sampling at that rate. In our experiments, *rmax* was 80 Hz and *r0* was found to be 67 Hz.

2. **Group Sampling**: The intuition for group sampling is that devices such as the CyberGrasp have different sensors that can be mapped to groups (e.g., all joints of a finger). We can isolate a sampling rate for each group and acquire data at different rates, based upon the group membership for each sensor. The advantage of this technique is its improvement over the fixed sampling technique by further reducing storage space and transmission requirements while maintaining accuracy. The difficulty in pursuing the grouped sampling strategy is in identifying the groups. Our intuition about natural groups may not be correct all the time.
3. **Adaptive Sampling**: This is a dynamic form of sampling where we try to find an optimum rate r_{ij} for each sensor *i* during a given window *j* of the session. The obvious advantage is the optimality of this approach. Adaptive sampling reduces bandwidth and storage requirements to far lower levels as compared to fixed or group sampling techniques. An additional benefit is that the sampling rate changes with the nature of the sessions. This makes the adaptive approach more efficient than the fixed or group sampling approach. The drawback is that it requires a complex implementation. We used a double buffering approach where a recording thread samples at the maximum rate possible and a storage thread performs the basic sampling methodology to identify the Nyquist sampling rates. This buffering approach means that some degree of real time acquisition is sacrificed.

The adaptive sampling approach looks particularly attractive because of its efficiency and robustness. In Shahabi et al. (2001), we provide details on these sampling techniques and the various tradeoffs among factors like bandwidth, storage, and computational complexity.

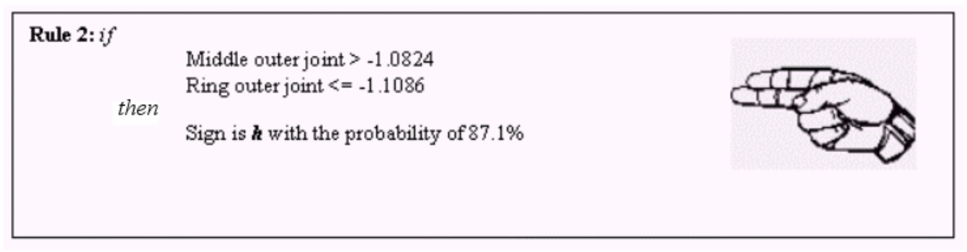
Classification Methods

In this chapter we explore three different classification techniques and evaluate the accuracy of each technique to detect 10 different hand signs. The employed techniques are C4.5 Decision Tree, Bayesian Classifier, and Neural Networks. Each classification technique is implemented in two different stages, the training phase and the recognition phase. We base our experiments on the data obtained from the first 22 sensors, as we believe that these sensor values are the most important and they hold all the information required in detecting a sign.

C4.5 Decision Tree

Tree induction methods are considered to be supervised classification methods that generate decision trees derived from a particular data set. C4.5 uses the concept of information gain to construct a tree of classificatory decisions with respect to a previously chosen target classification (Quinlan, 1993). The information gain can be described as the effective decrease in entropy resulting from making a choice as to which attribute to use and at what level. In addition, the output of the system is available as a symbolic rule base, which allows the system developer to determine the factors that have an impact on the selection strategy for a given application domain**.** C4.5 starts with large sets of cases belonging to known classes of data. The cases, described by any mixture of nominal and numeric properties, are scrutinized for patterns that allow the classes to be reliably discriminated. These patterns are then expressed as models, in the form of decision trees or sets of *if-then* rules, which can be used to classify new cases, with an emphasis on making the models understandable as well as accurate (Quinlan, 1993).

C4.5 consists of two major modules: decision tree maker and rule generator. At first the entire data set gets partitioned to a smaller subplace to construct a decision tree. However, for real world databases the decision trees become huge in practice. Large decision trees are always difficult to understand and interpret. In general, it is often possible to prune a decision tree to obtain a simpler and more accurate tree. However, a tree may not provide any significant insight into data. Figure 14-1 illustrates a rule generated by C4.5 in our experiment for the letter H.



Rule 2: *if*
Middle outer joint > -1.0824
Ring outer joint <= -1.1086
then
Sign is ***h*** with the probability of 87.1%

Figure 14-1: Sample rule for the letter "H."

We employed C4.5 Decision Tree because it provides a model to build a sign recognition language. In addition, decision trees in general and C4.5 in particular provide results as a set of understandable and interpretable rules. Finally, C4.5 has been used as a benchmark in several other papers on machine learning, artificial intelligence, and data mining.

C4.5 complexity is $O(nt)$ where t is the number of tree nodes, and the number of tree nodes often grows as $O(n)$ where n is the number of sessions. The complexity for non-numeric data would be $O(n^2)$, for numeric data $O(n^2 \log n)$, and for mixed-type data, somewhere in between.

Bayesian Classification

Bayesian Classifier is a fast-supervised classification technique. This method is able to reduce the risk of various hypotheses about the patterns of missing data. Furthermore, it results in both an accurate analysis of an incomplete database and decisions about the predictions from the data. Naïve Bayesian Classification performs well if the values of the attributes for the sessions are independent. Although this assumption is almost always violated in practice, recent work (Domingos & Pazzani, 1996) has shown that naïve Bayesian learning is remarkably effective in practice and difficult to improve upon systematically. Bayesian Classifier is suitable for large-scale prediction and classification tasks on complex and incomplete datasets. We have decided to use the naïve Bayesian Classifier in our application, for the following reasons. First, it is efficient for both the training phase and the recognition phase. Second, its training time is linear in the number of examples and its recognition time is independent of the number of examples. Finally, it provides relatively fine-grained probability estimates that can be used to classify the new session (Elkan, 1997).

The computational complexity of Bayesian Classification is fairly low as compared to other classification techniques. Consider a session with f attributes, each with v values. Then with the naïve Bayesian classifier with e sessions, the training time is $O(ef)$ and hence independent of v.

Neural Networks

We use Neural Networks for the recognition of both static and dynamic signs with a limited vocabulary. Supervised learning is being used for the classification. In this section, we first explain the basics of the Neural Network architecture that we used and then discuss the setting of its parameters for our experiments. An artificial neuron receives its inputs from a number of other neurons or from an external stimulus. A weighted sum of these inputs constitutes the argument to an activation function. This activation function is generally nonlinear (e.g., hard-limiting, sigmoid, or threshold logic). The resulting value of the activation function is the output of the Artificial Neural Network (ANN). This output gets distributed along weighted connections to other neurons. The actual manner in which the connections are made defines the flow of information in the network and is called the architecture of the network. Useful architectural configurations include single-layer, multilayer, feed-forward, feedback, and lateral connectivity. The method used to adjust weights in the process of training the network is called the learning rule. Artificial neural systems are not programmed, they are taught. The learning can be supervised (e.g., back-propagation) or unsupervised (e.g., self-organizing maps).

A Multilayer Perceptron (MLP) is trained using the supervised-learning rule. The most commonly used algorithm for such training is the error-back-propagation-algorithm. MLPs are feed-forward networks with one or more layers of nodes between the input and output layers of nodes. These additional layers contain hidden nodes that are not directly connected to both the input and the output nodes. The capabilities of the multilayer perceptrons stem from the nonlinearity used in these nodes. The number of nodes in the hidden layer must be large enough to form a decision region that can be as complex as required by a given problem. A three-layer perceptron can form arbitrarily complex decision regions. Hence, usually, most problems can be solved by three-layer (one hidden layer) perceptrons.

A vital attribute of any trained neural network is the ability to extract the discriminant information from a large number of examples. Hence, Neural Networks are ideal for complex pattern-recognition problems whose solution requires knowledge that is difficult to specify but which is available in the form of examples. They have been studied within a variety of applications.

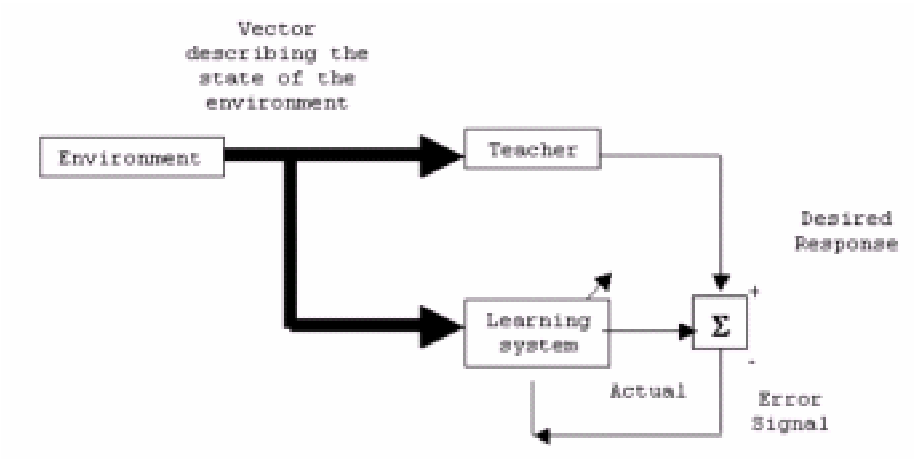


Figure 14-2: Supervised learning diagram with NN.

Supervised Learning Rules

Supervised learning is based upon the availability of an external teacher, as illustrated in Figure 14-2. The desired response represents the optimum action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the network emulate the teacher. When this condition is reached, we may remove the teacher and let the neural network deal with the environment thereafter entirely by itself. Figure 14-2 shows the supervised learning diagram. Examples of supervised learning algorithms include the back-propagation algorithm.

The Multilayer Perceptron Error BP Learning Algorithm

The back-propagation algorithm is an iterative algorithm designed to minimize the mean-squared error between the actual output of a feed-forward perceptron and the desired output. It requires continuous differentiable nonlinearity. The following equation assumes that a *sigmoid* logistic nonlinearity is used where the function $f(\beta)$ is:

$$f(\beta) = \frac{1}{(1+e^{(\varphi-\beta)})} \tag{14.1}$$

The MLP back-propagation algorithm can be described as follows (Lippmann, 1987). First, we initialize the weights and the offsets of the neural network to *small* random values. The next step involves presenting the neural network with the input and the desired outputs. We present a *continuous value input-vector x* (0), *x* (1), … *x* (n-1) and specify the desired outputs *d* (0), *d* (1), … *d* (*m*-1). Since we use the net as a classifier, all the outputs are set to 0, except for the output that corresponds to the class to which the input belongs, which is set to 1. The input could be new on each trial or samples from the training set could be presented cyclically until the weights stabilize. The complete set of training inputs is called an *epoch*. Next, we compute the actual outputs. We use the sigmoid nonlinearity from Equation 14.1 and use the following equation to compute the output:

$$Y\left[y(0), y(1), \ldots y(m-1)\right]:$$

$$X'(j) = f\left(\sum_{I=0}^{N-1}\left(w(i)(j) \; * x(j) \; - \; \phi(j)\right)\right) \text{ where } 0 \le j \le (N_1 - 1) \tag{14.2}$$

The output for the subsequent hidden layer and the final layer is then computed using a similar equation. We recursively adjust the weights, starting from the output node, working back towards the hidden layer. We use the following formula to adjust the weights:

$$w(i)(j)(t+1) = w(i)(j)(t) + (\mu * \S(j) * x'(i)) \tag{14.3}$$

In this equation *w(i)(j)(t)* is the weight from a hidden node *i* or from an input node *j* at time *t*, *x'(i)* is either the output from node *i* or is an input, μ is a gain term, called the Learning Rate, and *§(j)* is an error term for node *j*. We then compute the error using the following equation. Mean Square Error**:** $E = 1/2\left(\sum (d(j) - y(j))^2\right)$

If the error is sufficiently small, we stop with the learning process; otherwise, we iterate through the above processes until the error is sufficiently small.

While the algorithm works its way forward to the output layer, the error gradient is actually computed from the output layer, backwards. Hence, it is historically being called the *back-propagation* algorithm. The Neural Network convergence is sensitive to the number, type, and sequence of inputs during training and the initialization of random weights.

Implementation of Neural Network Classification over Static Signs

We used the following parameters throughout our experimentation. These parameters were chosen to be optimal in given conditions and given data, over multiple runs.

Number of Nodes:

1. Input Layer: 22 nodes (one per sensor), plus one threshold value node for the next layer.
2. Hidden Layer: The MLP used one hidden layer with 10 nodes.
3. Output Layer: 10 nodes, each corresponding to one posture.

Training Data:

With each of the 23 inputs (22 haptic glove values + 1 threshold) connected to each of the 11 hidden layer neurons (10 neurons + 1 threshold), and again each of these hidden layer neurons being connected to each of the 10 output neurons, the total number of weights in the network is $(23 \times 10) + (11 \times 10) = 340$ weights.

For our experiment on static signs with 340 weights, we establish the cardinality of the training set to achieve a good generalization as propounded in Vapnik and Chervonenkis (1971), approximately 10 times more, to cross the "*VCDim*" threshold. We analyzed the recorded log file for every subject-sign pair session and extracted 40 snapshots from each of these 10 subjects. Hence, we have a training data set of 10 subjects, each making 10 signs, and for each sign-subject pair we have 40 snapshots, resulting in 4000 sets of sensor values. We train the network for 500 epochs. The error rate stabilizes to two places after the decimal. We generate pseudorandom weights, the range of which is –1.0 to +1.0. The data is affected by noise but was input to the neural network without any preprocessing except for normalization. It was normalized to the range of –1 to +1. We strove to make the neural network learn on raw haptic data so that it learns to handle noisy data. This can be useful when we try to use the classifier in real-time immersive applications.

Similar work has been done in Salomon and Weissmann (1999), wherein they use all possible groupings of two fingers as input. This yields very good results on the training set, but the ability of this approach to be generalized needs to be ascertained. Our approach provides a good promise for an overall generalization.

Theoretical Setup for Classification of Dynamic Signs

We were preparing to classify a restricted vocabulary of dynamic signs. Each subject might perform the same sign with different speeds. With a fixed sampling rate (Shahabi et al., 2001) for each session, the chances were high that we would have the same sign represented by a different number of samples in different sessions. This called for a way to incorporate the *temporal* dimension into the haptic data. We proposed to use the Time-Delay Neural Network (TDNN) approach towards this end.

Time-Delay Neural Network

TDNN (Waibel, 1989) is a multilayer feed-forward network and it is trained with the back-propagation algorithm. We used TDNN for haptic data because it can learn and represent relationships between events in time and it can learn complex nonlinear decision surfaces, especially with high-dimensional input data. TDNN can learn inherent features in a manner that is invariant under translation in time. This can be achieved by feeding the input sequence into a tapped delay line, and then feeding the taps from the delay line into an MLP.

Input and its Preprocessing

Taking the time dimension into consideration results in a major bottleneck for this architecture, since we can unfold the sequence only over a finite period of time. The delay line can only be of finite extent, requiring the same number of sensor-value-sets for every input. We decided upon this fixed typical number 'N' as follows:

N = (typical length of a gesture in seconds $\times$ sampling rate)

With different subjects performing the signs with different speeds, it is not possible to have constant N for every sign at a fixed sampling rate. Hence we plan to implement standard signal processing reparameterization techniques such as Dynamic Time Warping (DTW), which has been implemented earlier in the speech recognition literature (Rabiner & Juang, 1993) or similar techniques on glove-based inputs, discussed in Sandberg (1997).

After expanding the temporal dimension of a gesture spatially, we decided on the length of the input *window* as 2 seconds. With a fixed sampling rate of q sessions/second, we shall have $2q$ as the length of delay line, giving $(2 \times q \times 22)$ input nodes.

Architecture

We planned to use two hidden layers and one output layer apart from the input layer. The number of units (nodes) in the output layer as well as the second hidden layer is V, where V is the size of the vocabulary. We planned to experiment with the exact number for the first hidden layer, though intuitively it seems that seven nodes to extract apparent features of a palm should be appropriate. Learning proceeds analogous to the back-propagation algorithm discussed earlier, although the optimal parameter settings will be different.

Performance Evaluation

We conducted several experiments to evaluate and compare our three different analysis techniques for ASL recognition: C4.5 Decision Tree, Bayesian Classifier, and Neural Networks. Below we explain our experimental setup, the results for each method and a comparison among these different techniques.

Experimental Setup

Fifteen subjects were selected to generate ASL signs from a given vocabulary. The subjects were asked to generate the following signs: A, B, C, D, E, F, G, H, I, and L, and data were stored in a database. The signs J and K are complicated and, taking the novice subjects into consideration, the signs were skipped for simplicity. We then determined the result of using each classification technique for ASL recognition. To evaluate each algorithm we used the cross-validation technique. We split the data into three sets, trained the system using two of the sets and conducted the tests using the third set. We implemented the test procedure in a round robin fashion and computed the average error (i.e., precision and recall). For example, if we split the data set into three different sets, denoted as *set-1, set-2* and *set-3*, we go through the following steps to perform the experiments in round robin and to compute the average error:

1. Train the data with *set-1* and *set-2* and test on *set-3*.
2. Train the data with *set-2* and *set-3* and test on *set-1*.
3. Train the data with *set-1* and *set-3* and test on *set-2.*

Storage of the Input

The Neural Network was trained using 4000 snapshots, as described earlier. This data was extracted from 100 session log files (10 subjects, 10 signs each, 40 different snapshots). The log files were produced as a result of recording the sessions.

For our experiments on static signs, we analyzed the recorded log files stored in a database and extracted the snapshot that has the sensor values consistent over a substantial period of time. To find such a snapshot, we used an SQL query similar to the one stated below. The following is an example of a simple SQL query when the database has only one snapshot and the generalization to more than one snapshot is straightforward. We assume the following table with attributes: CyberGlove (time, snapshot)

The following is a sample query:

```
SELECT snapshot FROM CyberGlove c1, CyberGlove c2
WHERE c2.snapshot IN(
SELECT max(time),snapshot FROM CyberGlove c3
WHERE c3.time < c2.time
ORDER by c3.time) AND
c1.snapshot - c2.snapshot < DELTA
ORDER BY (c1.snapshot - c2.snapshot)
```

Classification algorithms can be developed using incremental learning. The *learner* updates the rules, trees, and weights using the new session. The details of incremental learning are beyond the scope of this chapter and have been addressed in the machine learning literature. The details of the data acquisition techniques have been addressed earlier.

Results

Tables 14-5, 14-6, and 14-7 depict the precision and recall values for each of the classifiers. Figures 14-3 and 14-4 summarize the tables by comparing the average recognition error for each sign using the three different classification techniques.

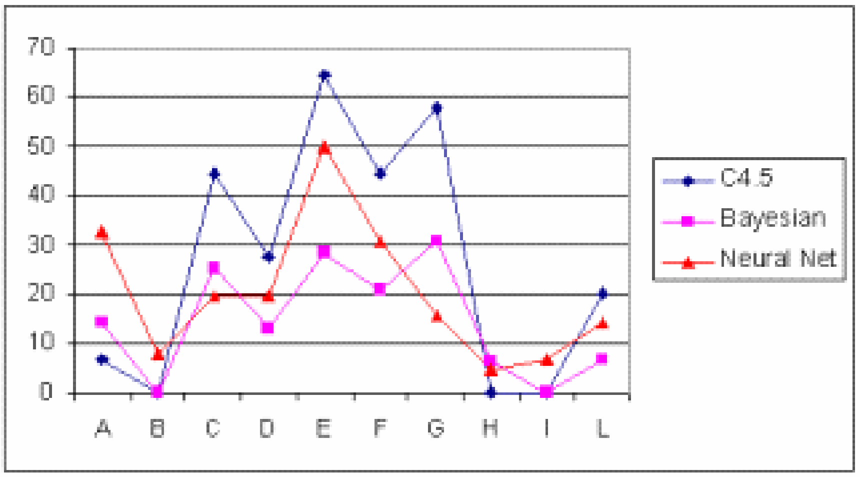


Figure 14-3: Sign recognition error.

	A	B	C	D	E	F	G	H	I	L
C4.5	6.6	0.0	44.4	27.7	64.4	44.4	57.7	0.0	0.0	20.0
Bayes-ian	14	0.0	25	13.06	28.4	20.83	30.52	6.4	0.0	6.4
Neural Net	32.8	7.87	19.6	19.5	50.04	30.55	15.73	4.9	6.67	14.24

Figure 14-4: Sign recognition error comparison.

The naïve Bayesian Classifier has the highest average accuracy with 50 training examples: 84.66% (with a standard deviation of 2.94). In contrast, C4.5 has an average of 78% (*SD* = 8) and Back-Propagation Neural Network has an average accuracy of 79.82% (*SD* = 7.92). Table 14-2 illustrates a comparison among the techniques.

Table 14-2: Overall classification error.

	Error	Standard Derivation
C4.5	22	8
Bayesian	15.34	2.94
Neural Net	20.18	7.92

Analysis

The Bayesian Classifier gives a very efficient and accurate result as compared to other classification techniques. The results of our experiments illustrate that C4.5 Decision Tree may not be suited to the task of sign recognition. Both Neural Networks and C4.5 have a large amount of variation in their performance. However, most often, C4.5 results are more interpretable and understandable. In contrast, the Neural Network architecture and procedure are not interpretable, and it is similar to a black box, in which case we only have access to input and output. Our experiments indicate that all of the classifiers performed relatively well on signs "B," "H," "I," and "L." Inspecting the signs, it appears that it was intuitive for subjects to perform these signs most consciously. Considering all the signs as points in a 22-dimension hyperspace, and computing the Euclidian distance among them, we realized that on the average, these four signs are quite apart from the rest of the signs, which justifies our observation. On the other hand, the letter "E" was quite close in distance to all the other signs and hence all classifiers were confused one way or the other with the recognition of letter "E."

The performance variation of individual classifiers over the signs can be traced back to the performance characteristics of each classifier. A neural network inherently tries to draw crisp distinguishing boundaries between groups of signs in the 22-dimensional hyperspace. Hence, it distinguished all the signs made when the hand is in the horizontal position (i.e., "C," "G," and "H") quite well (see Table 14-3). Note that although C4.5 was the best classifier for the letter "H," it had the minimum recognition error among the other letters with the neural net. With C4.5 and Bayesian Classifiers, the main assumption is that all features in a given space are independent. In general any strong dependency increases the level of error for both methods, while a low degree of dependency among features might have a negligible effect. Further, since C4.5 produces decisions based on a set of "if-then" rules, it tends to be relatively rigid, resulting in a high standard deviation as well as a high overall error. Since

Bayesian Classifier decides based on probability distribution of the input samples, it tends to perform quite well overall despite intuitive variations in performance of signs by different subjects.

Table 14-3: Best recognition technique for each sign.

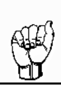
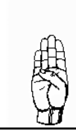
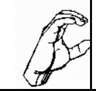
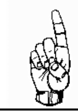
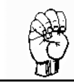
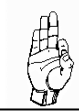
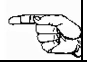
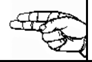
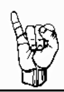
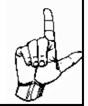
A	B	C	D	E	F	G	H	I	L
									
C4.5	C4.5, B	NN	B	B	B	NN	C4.5	C4.5, B	C4.5

Table 14-4: Nearest neighbors for each sign in multidimensional space.

	Nearest								Farthest	Avg. Euclidean Distance
A	E	G	L	D	I	H	C	F	B	2.284044
B	F	C	D	H	E	I	A	G	L	3.308474
C	D	F	E	I	A	B	H	L	G	2.192405
D	C	E	G	L	A	F	H	I	B	2.031382
E	A	D	C	I	G	F	L	H	B	2.179982
F	C	B	D	E	I	A	H	G	L	2.578171
G	L	A	H	D	E	I	C	F	B	2.441150
H	G	D	A	L	E	C	F	I	B	2.627276
I	A	E	C	D	F	G	L	H	B	2.693604
L	G	D	A	E	H	C	I	F	B	2.697530

As illustrated in Table 14-8,[1] we see that the best classifier for a sign is not necessarily the one that confuses the sign with fewer other signs. The decision to choose a classifier from given classifiers becomes very much application-dependent. To illustrate this observation consider an example: C4.5 gives the highest average accuracy for sign L, but the other two techniques confuse L with fewer signs (one each) compared to that of C4.5 (three signs). In sum, we show that even with a small pool of snapshots, with a fast learner such as Naïve

[1]This table has been created using the previous tables. The percentage error > 0.00 for each sign is taken as confusion of that sign for the classifier. For Neural Networks, this threshold is fixed at 3.00.

Bayesian Classifier and an appropriate I/O design we can achieve an acceptable performance.

Table 14-5: C4.5 Precision and recall.

	A	B	C	D	E	F	G	H	I	L
A	93.33	0.00	0.00	6.66	00.00	0.00	00.00	0.00	00.00	0.00
B	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C	6.66	0.00	66.66	6.66	6.66	6.66	6.66	0.00	0.00	0.00
D	0.00	0.00	20.00	73.33	0.00	0.00	0.00	0.00	0.00	6.66
E	6.66	6.66	0.00	13.33	46.66	13.33	13.33	0.00	00.00	0.00
F	0.00	0.00	6.66	0.00	20.00	66.66	6.66	0.00	0.00	0.00
G	0.00	0.00	0.00	26.66	6.66	0.00	53.33	00.00	0.00	13.33
H	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
L	0.00	0.00	0.00	6.66	6.66	0.00	6.66	0.00	0.00	80.00

Table 14-6: Bayesian precision and recall.

	A	B	C	D	E	F	G	H	I	L
A	86.00	0.00	0.00	0.00	14.00	0.00	00.0	0.00	00.00	0.00
B	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C	0.00	0.00	75.00	20.00	0.00	5.00	0.00	0.00	0.00	0.00
D	0.00	0.00	10.70	86.04	3.28	0.00	0.00	0.00	0.00	0.00
E	0.00	0.00	0.00	7.10	71.60	21.30	0.00	0.00	00.00	0.00
F	0.00	0.00	0.00	0.00	14.0	86.00	0.00	0.00	0.00	0.00
G	18.3	0.00	0.00	0.00	0.00	6.11	69.50	6.11	0.00	0.00
H	0.00	0.00	0.00	0.00	0.00	0.00	6.40	93.6	0.00	0.00
I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
L	0.00	0.00	0.00	0.00	0.00	0.00	6.40	0.00	0.00	93.60

Table 14-7: Neural network precision (standard deviation) and recall.

	A	B	C	D	E	F	G	H	I	L
A	71.08, 15.78	1.65, 5.36	2.59, 5.62	3.90, 8.59	4.12, 10.56	3.92, 7.77	3.22, 4.67	3.00, 4.27	3.49, 7.20	2.10, 2.84
B	1.86, 4.20	93.49, 8.46	1.67, 3.02	0.57, 1.83	1.24, 2.74	0.53, 1.14	0.00, 0.00	0.12, 0.51	0.00, 0.00	0.00, 0.00
C	1.02, 3.44	9.00, 10.38	80.49, 10.17	7.12, 9.57	0.39, 2.77	0.92, 2.60	0.00, 0.00	0.41, 1.36	0.16, 1.11	0.04, 0.28
D	1.14, 5.54	1.25, 4.15	8.00, 10.66	82.08, 16.65	4.04, 8.50	2.98, 7.09	0.00, 0.00	0.25, 1.03	0.00, 0.00	0.00, 0.00
E	3.80, 8.16	0.20, 0.84	4.53, 10.62	11.63, 13.18	48.80, 21.13	11.49, 18.20	2.63, 6.84	2.45, 7.19	13.63, 21.59	0.22, 1.27
F	6.98, 11.13	0.96, 3.97	7.10, 12.74	1.18, 3.80	6.14, 9.57	75.14, 23.88	0.00, 0.00	0.80, 2.52	1.22, 3.97	0.06, 0.24
G	0.82, 3.72	1.49, 3.46	1.67, 3.56	3.61, 7.27	1.57, 5.38	0.00, 0.00	81.00, 17.87	1.65, 4.31	0.00, 0.00	7.94, 12.94
H	1.20, 4.14	1.06, 3.05	0.55, 1.83	1.29, 4.09	0.02, 0.14	0.12, 0.70	0.00, 0.00	95.51, 7.60	0.00, 0.00	0.00, 0.00
I	0.57, 2.72	0.67, 1.88	2.35, 4.67	1.43, 4.64	0.00, 0.00	0.29, 1.00	0.00, 0.00	0.35, 1.86	94.12, 9.11	0.00, 0.00
L	1.55, 4.46	0.78, 2.12	1.53, 3.33	0.88, 2.87	0.51, 2.55	0.24, 1.41	4.39, 8.20	0.86, 2.46	0.33, 1.42	88.57, 9.32

Table 14-8: Number of other signs with which each sign is confused for different classifiers.

Sign	Bayesian	C4.5	Neural Networks
A	1	1	5
B	0	0	0
C	2	5	2
D	2	2	2
E	2	5	5
F	1	3	3
G	3	3	2
H	1	0	0
I	0	0	0
L	1	3	1

RELATED WORK

Various research groups worldwide have been investigating the problem of sign recognition. We are aware of the two main approaches. The machine-vision-based approaches analyze the video and image data of a hand in motion. This includes both the 2D and 3D position and the orientation of one or two hands. The haptics-based approaches analyze the haptic data from a glove. Quantified values of the various degrees of freedom for the hand constitute the data. These efforts have resulted in the development of devices such as the CyberGlove.

We categorize the first approach based on the techniques employed. Darrell and Pentland (1993) discuss vision-based recognition with "Template Matching." Heap and Samaria (1995) employ Active Shape models. Several studies such as Martin and Crowley (1997) and Birk, Moeslund, and Madsen (1997) propose to use Principal Components Analysis. Yet another method of recognition using linear fingertips is described in Davis and Shah (1993). Banarase (1993) uses a neocognitron network. Like our group, various researchers have tried to recognize various sign languages all over the world using different methods. These include the American (ASL), Australian (AUSLAN), Japanese (JSL), and Taiwanese (TWL) Sign Languages, to name a few. As for the most relevant, the task of ASL recognition has been pursued by numerous research groups (Starner, 1995; Starner & Pentland, 1996; Vogler & Metaxas, 1997, to cite a few who use the Hidden Markov Models). An excellent survey of vision-based sign-recognition methods is provided in Wu and Huang (1999).

Using gloves and haptic data, Fels and Hinton (1995) employed a VPL Glove to carry out sign recognition. Takahashi and Kishino (1991) also investigated the understanding of the Japanese Kana manual alphabet (consisting of 46 signs) using a VPL DataGlove. They constructed a table that designated the positions of individual fingers and joints that would indicate a particular hand shape. They reported that they could successfully interpret 30 of the 46 signs, while the remaining 16 could not be reliably identified, due to a variety of constraints, such as the fact that they were moving gestures and that sufficient distinction could not be made in situations where fingertips touched. Sandberg (1997) provides an extensive coverage and employs a combination of Radial Basis Function Network and Bayesian Classifier to classify a hybrid vocabulary of static and dynamic hand signs. One more variant exists, Recurrent Neural Networks, used by Murakami and Taguchi (1991) for classifying Japanese sign language. We believe that the work by Salomon and Weissmann (1999) is the most relevant to our own, since it attempts to recognize signs using the same CyberGlove and back-propagation algorithm that we use. Hidden Markov Models are popular here too, which is reflected in Nam and Wohn (1996) and Lee and Yangsheng (1996). The latter is particularly relevant because it presents an application for learning signs through Hidden Markov Models, taking the data input from the CyberGlove. Wu, Wen, Yibo, Wei, and Bo (1998) use a combination of MLP-BP and HMM. Kadous (1995) used instance-based learning to classify Australian Sign Language. Newby (1993) studied glove-based template matching using a simple sum-of-squares approach. Rubine (1991) proposed feature extraction, while Charaphayan and Marble (1994) compared the approaches of Dynamic Programming, HMM, and Recurrent Neural Networks. Dorner and Hagen (1994) are unique in that they have taken a holistic approach to the question of ASL interpretation. Hagen's work involved building a deductive database that successfully translates from a standardized form of ASL into spoken English. Other neural network algorithms—Radial Basis Function Network (RBFN), Orthogonal Least Squares and Self-Organizing Maps—have also been tested on various kinds of data-glove inputs, in Lin (1998) and Ishikawa & Matsumura (1999). Salomon and Weissmann (2000) go further to show that RBFN is better than MLP-BP for classifying dynamic signs in an evolutionary manner.

Our work is distinct from all of the above-mentioned works because we provide a complete system including I/O unit, data acquisition module, database structure, and classification methods for ASL recognition. All our analysis is carried out on raw haptic data. We are the first to use Decision Tree for the analysis of haptic data. We are also the first to use Bayesian Classifier for *raw* (i.e., not preprocessed) haptic data analysis. Taking our framework into consideration, we are also the first to use Back-Propagation Neural Networks for the recognition of static signs. We propose to use Time-Delay Neural Networks for the recognition of dynamic signs.

CONCLUSION AND FUTURE WORK

In this chapter, we analyzed three different classification techniques for sign language recognition. We showed that Decision Tree, Bayesian Classifier, and Neural Networks could be used for ASL recognition. Bayesian Classifier proved to be the fastest classification technique among the three we evaluated. It also proved to have the best classification accuracy for static sign recognition. We carried out several preliminary experiments and the results of our experiments suggest that Bayesian Classifier can be used to develop a real-time sign language recognition system. However, more work needs to be carried out in order to establish the validity of our results, which are very encouraging in the early stages of experimentation. There are many open questions, obstacles, and problems that need to be dealt with before we achieve an efficient, reliable, and applicable ASL recognition system.

We intend to extend our work in several ways. First, in addition to Time-Delay Neural Networks, we also intend to investigate Evolving Fuzzy Neural Networks for the recognition of dynamic signs. It would be interesting to compare the effectiveness of these two techniques. Second, we want to use the analysis of haptic data that we pursued for the modeling of haptic data in a database. Our analysis would tell us what data we need to store at which level of abstraction for a given application. Third, we would like to analyze haptic data at the third level of abstraction, which requires us to analyze preprocessed haptic data. Finally, we propose to use shape recognition techniques for the recognition of dynamic signs based upon a fixed sign language vocabulary. Here, each dynamic sign would be considered to have a static part and an associated dynamic part. Using a shape to represent the dynamic part would let us view the dynamic sign in a time-independent manner. Such an approach would need an efficient technique for tracking the haptic device.

Acknowledgments

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and ITR-0082826, NASA/JPL contract number 961518, DARPA and USAF under agreement number F30602-99-1-0524, and unrestricted cash/equipment gifts from NCR, IBM, Intel and SUN.

References

Banarase, D. (1993). *Hand posture recognition with the neocognitron network*. Technical report, School of Electronic Engineering and Computer Systems, University College of North Wales.

Birk, H., Moeslund, T., & Madsen, C. (1997). Real-time recognition of hand alphabet gestures using Principal Component Analysis. In *Proceedings of The 10th Scandinavian Conference on Image Analysis*. Lappeenranta, Finland**.**

Cracknell, J., Cairns, A., Gregor, P., Ramsay, C., & Ricketts, I. (1994). Gesture recognition: An assessment of the performance of recurrent neural networks versus competing techniques. *IEE Colloquium on Applications of Neural Networks to Signal Processing*, Digest No. 1994/248, 8/1–8/3.

Darrell, T., & Pentland, A. (1993). *Recognition of space-time gestures using a distributed representation*. Technical Report No.197, MIT Media Laboratory Vision and Modeling Group.

Davis, J., & Shah, M. (1993). *Visual gesture recognition*. Technical report, University of Central Florida.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 105–112). Bari, Italy.

Dorner, B., & Hagen, E. (1994). Towards an American Sign Language interface. *Artificial Intelligence Review*, *8*, 235–253.

Elkan, C. (1997). Boosting and naive Bayesian learning. In *Proceedings of International Conference on Knowledge Discovery in Databases*. Newport Beach, CA.

Fels, S., & Hinton, G. (1995). Glove-talk II: An adaptive gesture-to-format interface. In *Proceedings of Conference on Human Factors in Computing Systems*. Denver, CO.

Heap, A., & Samaria, F. (1995). Real-time hand tracking and gesture recognition using smart snakes. *Olivetti Research Limited Tech Report*, *95*(1).

Ishikawa, M., & Matsumura, H. (1999). Recognition of a hand-gesture based on self-organization using a DataGlove. In *Proceedings of the Sixth International Conference on Neural Information Processing*, Vol. 2 (pp. 739–745). Kobe, Japan.

Kadous, W. (1995). *Grasp: Recognition of Australian Sign Language using instrumented gloves*. Bachelor's thesis, University of New South Wales, Australia.

Lee, C., & Yangsheng, X. (1996). Online interactive learning of gestures for human/robot interfaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 4 (pp. 2982–2987).

Lin, D. (1998). Spatio-temporal hand gesture recognition using neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 3 (pp. 1794–1798). Anchorage, AK.

Lippmann, R. (1987). An introduction to computing with Neural Nets. *IEEE Acoustics, Speech and Signal Processing Magazine*, *4*(2), 4–22.

Martin, J., & Crowley, J. (1997). An appearance-based approach to gesture recognition. In *Lecture Notes in Computer Science*, Vol. 1311 (pp. 340–347). Springer Verlag.

Murakami, K., & Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In *Proceedings of Human Factors in Computing Systems* (pp. 237–242). New Orleans, LA.

Nam, Y, & Wohn, K. (1996). Recognition of space-time hand-gestures using Hidden Markov Model. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology* (pp. 51–58). Hong Kong.

Newby, G. (1993). Gesture recognition using statistical similarity. In *Proceedings of Virtual Reality and Persons with Disabilities*. Northridge, CA.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann.

Rabiner, L., & Juang, B. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall.

Rubine, D. (1991). Specifying gestures by example. *Computer Graphics*, *25*(3), 329–337.

Salomon, R, & Weissmann, J. (1999). Gesture recognition for virtual reality applications using data glove and neural networks. In the *Proceedings of IEEE International Joint Conference on Neural Networks*, Washington, DC.

Salomon, R., & Weissmann, J. (2000). Evolutionary tuning of neural networks for gesture recognition. In *Proceedings of the Congress on Evolutionary Computation*, Vol. 2 (pp. 1528–1534). San Diego, CA.

Sandberg, A. (1997). *Gesture recognition using neural networks*. Master's thesis TRITA-NA-E9727, Royal Institute of Technology, Sweden.

Shahabi, C., Barish, G., Ellenberger, B. , Jiang, N., Kolahdouzan, M., Nam, A.,& Zimmermann, R. (1999). *Immersidata management: Challenges in management of data generated within an im-*

mersive environment. Paper presented at Fifth International Workshop on Multimedia Information Systems (MIS99), Palm Springs, CA.

Shahabi, C., Barish, G., Kolahdouzan, M., Yao, D., Zimmermann, R., Fu, K., & Zhang, L. (2001). Alternative techniques for the efficient acquisition of haptic data. In *Proceedings of ACM SIGMETRICS/Performance 2001*. Cambridge, MA.

Starner, T. (1995). *Visual recognition of American Sign Language using Hidden Markov Models*. Master's Thesis, Program in Media Arts & Sciences, MIT Media Laboratory.

Starner, T., & Pentland, A. (1996). *Real-time American Sign Language recognition from video using Hidden Markov Models*. Technical report, MIT, 1996.

Takahashi, T., & Kishino, F. (1991). Hand gesture coding based on experiments using a hand gesture interface device. *ACM SIGCHI Bulletin*, *23*(2), 67–74.

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of Probability and Its Applications*, Vol. 16 (pp. 262–280).

Vogler, C., & Metaxas, D. (1997). Adapting Hidden Markov Models for ASL recognition by using three-dimensional computer vision methods. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Orlando, FL.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1989). Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, *37*(3), 328–339.

Wu, J., Wen, G., Yibo, S., Wei, L., & Bo, P. (1998). A simple sign language recognition system based on data glove. *In Proceedings of Fourth International Conference on Signal Processing*, Vol. 2 (pp. 1257–1260). Beijing, China.

Wu, Y., & Huang, T. (1999). Vision-based gesture recognition: A review. In *Proceedings of the 3rd Gesture Workshop* (pp. 103–115). Gif-sur-Yvette, France.