# Device Independence and Extensibility in Gesture Recognition*

Jacob Eisenstein, Shahram Ghandeharizadeh, Leana Golubchik,
Cyrus Shahabi, Donghui Yan, Roger Zimmermann
Department of Computer Science
University of Southern California
Los Angeles, CA 90089, USA

## Abstract

*Gesture recognition techniques often suffer from being highly device-dependent and hard to extend. If a system is trained using data from a specific glove input device, that system is typically unusable with any other input device. The set of gestures that a system is trained to recognize is typically not extensible, without retraining the entire system. We propose a novel gesture recognition framework to address these problems. This framework is based on a multi-layered view of gesture recognition. Only the lowest layer is device dependent; it converts raw sensor values produced by the glove to a glove-independent semantic description of the hand. The higher layers of our framework can be reused across gloves, and are easily extensible to include new gestures. We have experimentally evaluated our framework and found that it yields at least as good performance as conventional techniques, while substantiating our claims of device independence and extensibility.*

## 1 Introduction

Gesture recognition offers a new medium for human-computer interaction that can be both efficient and highly intuitive. However, gesture recognition software is still in its infancy. While many researchers have documented methods for recognizing complex gestures from instrumented gloves at high levels of accuracy [6, 8, 11, 13], these systems suffer from two notable limitations: device dependence and lack of extensibility.

Conventional approaches to gesture recognition typically involve training a machine learning system to classify gestures based on sensor data. A variety of machine learning techniques have been applied, including hidden Markov models [6, 8, 14], feedforward neural networks [13], and recurrent neural networks [6, 10]. These different approaches have a common feature: they all treat gesture recognition as a one-step, *single-layer* process, moving directly from the sensor values to the detected gesture. Consequently, the properties of the specific input device used in training are built into the system. For example, a system that was trained using a 22 sensor CyberGlove would almost certainly be of no use with a 10 sensor DataGlove. The system expects 22 inputs, and would be unable to produce meaningful results with the 10 inputs offered by the DataGlove.

Ideally, a gesture recognition system should be able to work with a variety of input devices. As the number of sensors is reduced, the performance might degrade, but it should degrade gracefully. We call this property *device independence*, and it is the first significant advantage of our approach.

In order to achieve device independence, we have reconceptualized gesture recognition in terms of a *multi-layer* framework. This involves generating a high-level, device-independent description of the sensed object – in this case, the human hand. Gesture recognition then proceeds from this description, independent of the characteristics of any given input device.

Because our approach allows the gesture recognition system to use a clean, semantic description of the hand, rather than noisy sensor values, much simpler techniques can be employed. It is not necessary to use anything as complicated as a neural network; rather, simple template matching is sufficient. Template matching provides a key advantage over more complex approaches: it is easily extensible, simply by adding to the list of templates. To recognize a new gesture with a conventional system, the entire set of gestures must be relearned. But we will show experimentally that with our approach, it is possible to add new gestures without relearning old ones. These new gestures are recognized with nearly the same accuracy as those in the original training set. Thus, *extensibility* is the second main advantage of our approach.

*This research is supported in part by NSF grants EEC-9529152 (IMSC ERC), and IIS-0091843 (SGER).

In Section 2, we describe our multi-layer framework in more detail. Section 3 presents our implementation for the task of ASL fingerspelling recognition, which we believe will generalize to other virtual reality applications. The results of an experimental evaluation of this implementation are given in Section 4. Section 5 surveys related work, and Section 6 presents brief conclusions and future research directions.

## 2 A Multi-Layer Framework

Our proposed framework is based on a multi-level representation of sensor data. It consists of the following four levels:

1. Raw data: This is the lowest layer and contains continuous streams of data emanating from a set of sensors. We have addressed the analysis of this general class of data in [4]. In this paper, we are specifically concerned with data generated by the sensors on a glove input device. A detailed description of the sensors included with the CyberGlove is found in [7]. The data at this level is highly device-dependent; each device may have a unique number of sensors, and the sensors may range over a unique set of values. However, raw sensor data is *application* independent; no assumptions are made about how the data will be used, or even what it describes. Conventional approaches to dealing with streaming sensor data typically operate at exactly this level. Consequently, these approaches are usually very difficult to adapt to new devices, and they fail to take advantage of human knowledge about the problem domain.

2. Postural Predicates: This level contains a set of predicates that describe the posture of the hand. Table 1 provides a list of the predicates to represent the hand postures for ASL fingerspelling. A vector of these predicates consisting of 37 boolean values describes a general hand posture. For example, a *pointing* posture is described by noting that the index finger is open, Open (index_finger), while every other finger is closed, Closed (thumb), Closed (middle_finger), etc. Each predicate describes a single features of the overall posture – e.g., Closed (index_finger). In our pointing posture, five vector values (corresponding to the aformentioned predicates evaluate as *true*), while the remaining ones evaluate as *false*.

   While we do not claim that Table 1 presents a comprehensive list that captures all possible postures, we do show that these predicates can describe the ASL alphabet. Preliminary research on other fingerspelling systems suggests that this set of predicates is general, and we plan to investigate its generality in support of non-fingerspelling applications. To this end, we plan to apply this set of predicates to a virtual reality application in future research.

   The postural predicates are derived directly from the lower level of raw sensor data. This process is described in more detail later in Section 3.2. The derivation of postural predicates from sensor data is, by necessity, device-dependent. However, it is application-independent, if our postural predicates are indeed general over a range of applications.

   Once postural predicates are extracted from the sensor data, device-independent applications can be implemented using this higher level of representation. Thus, our multi-layered approach provides for the sound software engineering practice of modular reuse. Some modules can be reused across multiple devices, while others can be reused across multiple applications.

3. Temporal Predicates: Our work thus far has mainly focused on postural predicates. Here, we assume the ASL alphabet as our working data set where most signs are $static$ and do not require any hand motion (and hence have no temporal aspect). The extension of the framework to temporal signs is part of our future work.

4. Gestural Templates: This layer contains a set of templates, each of which corresponds to a whole hand gesture. Postures contain no temporal information; a gesture may contain temporal information, although this is not required. A gesture is a description of the changing posture and position of the hand over time. An example of a gesture is a hand with a pointing index finger moving in a circular trajectory with a repetitive cycle.

   Gestures are described as *templates* because they are represented as a vector of postural and temporal predicates. In order to classify an observed hand motion as a given gesture, the postural and temporal predicates should match the gestural template. This might be an approximate match; in our ASL application, we simply choose the gestural template that is the closest match to the observed data (see Section 3.1.1 for details).

## 3 Implementation

Figure 1 shows the two key modules of our implementation: 1) a set of predicate recognizers, and 2) a

| Name | Definition | Applicability | Number of predicates |
|---|---|---|---|
| Open (X) | Indicates that finger X is extended parallel to the palm. | Any finger, and the thumb | 5 |
| Closed (X) | Indicates that finger X is closed with the palm. Note that the *open* and *closed* predicates are mutually exclusive, but they are not complete. A finger may neither entirely open, nor closed. | Any finger, and the thumb | 5 |
| Touching-thumb (X) | Indicates that finger X is touching the thumb. | Any finger other than the thumb | 4 |
| Grasping (X) | Indicates that finger is grasping something with the thumb. | Any finger other than the thumb | 4 |
| Split (X, Y) | Indicates that adjacent fingers X and Y are spread apart from each other. | Any adjacent pair of fingers, and the index finger and the thumb. | 4 |
| Crossing (X, Y) | Indicates that finger X is crossing over finger Y, with Y closer to the palm than X. | Applies to any two fingers, but finger crossings that are very difficult (e.g. pinky over index finger) are not included. | 14 |
| Palm-facing-in () | Indicates that the palm is facing the signer, rather than the recipient of the signs. | Applies to the whole hand. | 1 |

**Table 1. Postural Predicates**

| Alphabet | Set of predicates (corresponding to a template) |
|---|---|
| A | Closed (F1, F2, F3, F4) |
| B | Open (F1, F2, F3, F4), Closed (T) |
| C | Grasping F1, F2, F3, F4 (1in, 0 degrees) |
| D | Open (F1), Touching-thumb (F2, F3) |
| G | Open (F1), Closed (F2, F3, F4), Crossing (T, F2), Palm-facing-in() |
| T | Closed (F2, F3, F4), Crossing (T, F2), Crossing (F1, T) |

**Table 2. Some ASL Fingerspelling Templates**

template matcher. The predicate recognizers convert raw sensor data to a vector of predicates. The template matcher then identifies the nearest gestural template. The template matcher is assisted by two other components: a *confidence* vector, and a probabilistic *context*. We will first describe how these components work together to detect ASL signs. Next, Section 3.2 describes how the system is trained.

## 3.1 ASL Sign Detection

This section explains how the system moves from sensor data to the vector of predicates. We then describe the basic template matching technique, and show how it can be augmented with context and the confidence vector.

### 3.1.1 Predicate Recognizers

The predicate recognizers use traditional gesture recognition methods to evaluate each postural predicate from a subset of the sensor data. In this case, we implemented the predicate recognizers as feedforward neural networks; we have explored other approaches in the past [3]. Each predicate recognizer need not consider data from the entire set of sensors. Rather, the sensors are mapped as input to the predicate recognizers manually, using human knowledge of which sensors are likely to be relevant to each predicate. For example, the predicate Crossing (T, F1) receives input only from the sensors on the thumb and index finger. By mapping only those sensors that are relevant to each predicate recognizer, human knowledge can be brought to bear to dramatically improve both the efficiency and accuracy of training. Table 1 shows the six predicate types and the thirty-seven predicates required to describe a single handshape.

To perform recognition of these thirty-seven postural predicates, we employ thirty-seven individual neural networks (see Figure 1). Each neural net consumes between 4 and 10 sensor values, includes ten hidden nodes, and produces either a zero or a one as output, denoting the logical valence of its predicate. The outputted predicates are then collated together into a vector, which is fed as input to a template matcher.

### 3.1.2 Template Matching

The gesture recognizers for a specific application, such as ASL, are realized using these postural predicates. Since these gesture recognizers manipulate high-level *semantic* data rather than low-level sensor values, it be-

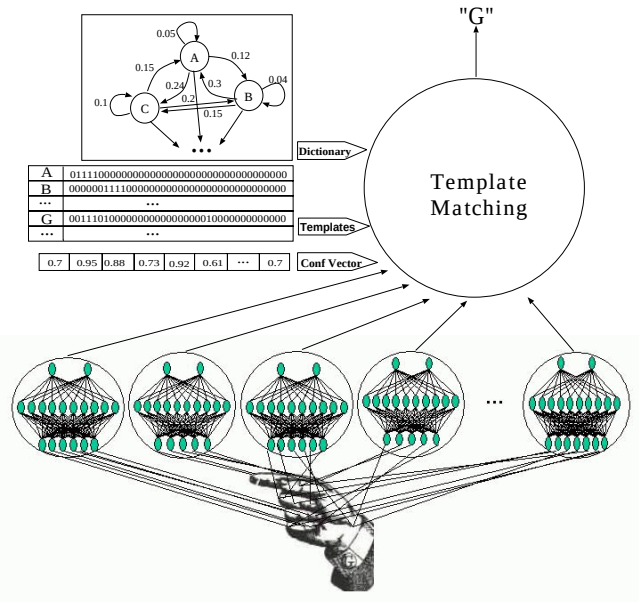


**Figure 1. A Multi-Layer Framework to Detect Static ASL Signs**

comes possible to employ simpler and more extensible approaches. Our system performs gesture recognition by simple template matching on the detected vector of postural predicates. Template matching can be extended by simply adding to the set of pre-existing templates. In addition, this template matching component can be used across many different gloves (see Section 4).

The template matcher works by computing the Euclidean distance between the observed predicate vector and every known template. The template that is found to be the shortest distance from the observed predicate vector is selected. Mathematically, for a perceived predicate vector $v$, we want to find the gesture template $i$ that minimizes $d_{i,v}$, which is the Euclidean distance between the two vectors.

$$d_{i,v} = \sum_{0 \leq p < P} |\mathbf{i}[p] - \mathbf{v}[p]| \quad (1)$$

$P$ is equal to the total number of predicates; in our application $P = 37$. Sections 3.1.2 and 3.1.3 will augment this equation with context and confidence to improve performance.

Table 2 shows the true predicates in the template for several of the static signs in the ASL fingerspelling alphabet. For a complete list of the predicates for every fingerspelling sign in ASL, see [2]. They were determined by consulting an ASL textbook. Since the meaning of each predicates is entirely straightforward, constructing the templates is not expected to be a significant bottleneck to the usage or extension of our system. We believe that manually specifying new templates is far easier than finding additional training data and retraining a neural network.

### 3.1.3 Confidence

Ideally, each predicate recognizer performs perfectly; failing that, we would at least like to see all of the predicate recognizers perform equally well. In reality, this is not the case. For example, the Closed(T) predicate recognizer might correctly classify every test case, while the Crossing(T, F1) predicate recognizer might produce results no better than chance. To treat these two predicate recognizers equally would be a mistake; we should apply more weight to the data from the Closed(T) recognizer, and ignore data from the Crossing(T, F1) recognizer. To achieve this, we construct a vector **k**, with *confidence* ratings between 0 and 1 for every predicate recognizer. The template matcher uses the confidence vector by factoring it into the distance metric:

$$d_{i,v} = \sum_{0 \leq p < P} \mathbf{k}[p]|\mathbf{i}[p] - \mathbf{v}[p]| \quad (2)$$

For example, suppose the predicate recognizers return the vector “0110”, and the two closest templates are “1110” and “0111”. Suppose that it is known that the first predicate recognizer (first bit) is only as good as chance, with a confidence rating of 0.1, but the 4th predicate recognizer (fourth bit) is rated with a confidence of 0.9. In this case, it is very likely that the fourth bit is indeed zero, while the value of the first bit is uncertain. Equation 2 uses this information, and selects “1110” as the appropriate template.

### 3.1.4 Context

Language users typically follow known patterns, rather than producing letters or words at random. Using context, the system constrains the space of possible utterances to improve performance. Context also reduces the effect of noise on the data, and can act as a tie-breaker between two templates that are equally close to the observed predicate vector.

We incorporate context into our application using an n-gram letter model [9, 16]. An n-gram is a probabilistic model of sequences of symbols, which we can use to predict the next symbol. For example, suppose that we are using trigrams (n = 3), and we know that the current context is “_q”: we are at the beginning of the word, and the first letter is q. Our trigram model will tell

us that the trigram "_qu" is far more probable than any other trigram, given this context. Thus, we can form a strong expectation that the next letter will be "u". Even if noisy data obscures the recognition, the context component can help us make the correct identification.

Context is represented by a vector **c**, which includes the recent history of signs produced by the user. The size of **c** is equal to the size of the n-gram minus one; if trigrams are used, then **c** stores a two-element history. We then use the n-gram probabilities to find the *conditional probability* that each possible letter follows the letters in the context. The conditional probability of some letter $i$, given the context **c**, is denoted by $P(i|\mathbf{c})$.

We now have to factor this probability into our original equation. Recall that originally we chose the gesture template $i$ that minimizes $d_{i,v}$ for an observed predicate vector $v$. This is the same thing as choosing the gesture template $i$ that maximizes $1/d_{i,v}$. We can include the conditional probability of each gesture in the equation by instead maximizing:

$$P(i|\mathbf{c}) \left( \frac{1}{d_{i,v}} \right)^n \tag{3}$$

By varying $n$, we can control the extent to which context influences the gesture recognition. As $n$ approaches zero, the $(1/d_{i,v})^n$ term approaches 1, and the conditional probability becomes more important. As $n$ goes to infinity, the $(1/d_{i,v})^n$ term becomes more significant, overwhelming the conditional probability and dominating the calculation. This tradeoff is quantified in detail in [2].

## 3.2 Training

The training phase impacts two components of the system: 1) each of the 37 predicate recognizers, and 2) the confidence vector. We describe each in turn.

### 3.2.1 Training the Gesture Recognizers

Figure 2 shows how the predicate recognizers are trained. During training, the system is presented with a set of example inputs, and the corresponding expected output templates. While the sensor values are passed along to the appropriate predicate recognizers, the expected output is passed to the list of gesture templates. The appropriate template is selected; for example, if the expected output is a "G," then the template "0110...1" is selected. The output of the i$^{th}$ predicate recognizer must match the i$^{th}$ element in the template vector. In our example, the output of the 2$^{nd}$ predicate must be a "1." If they do not match then the expected value is returned

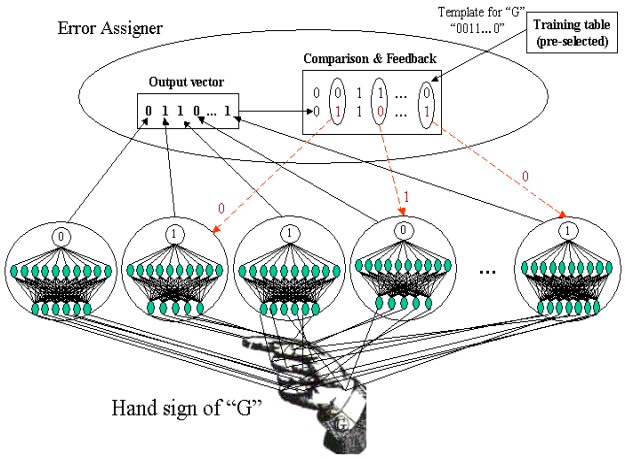


**Figure 2. Training of Spatial Sign Detector**

to the predicate recognizer and it is trained using a conventional error backpropagation method for feedforward neural networks [1, 12].

While the use of thirty-seven individual neural networks might appear computationally expensive, the small size of each network makes the cost of this approach similar to that of traditional techniques. The average predicate recognizer has six inputs, ten hidden nodes, and one output, for a total of $6 * 10 + 10 * 1 = 70$ weights that must be updated during training. With 37 networks, a total of $37 * 70 = 2590$ weights must be updated. A traditional approach, employing a single large neural network to recognize ASL signs directly from raw sensor data, would require 22 inputs (one for each sensor), and 24 outputs (one for each static sign). In our experiments, we found that 25 hidden nodes yielded the best performance for the traditional single-network approach. Thus, this approach still requires that $22 * 25 + 25 * 24 = 1150$ weights be updated. Even though we use thirty-seven networks instead of one, our approach is only a little more than two times as costly as the conventional technique. We believe that the benefits of device-independence and extensibility, which will be quantified later, more than justify this additional cost. At most, 3000 iterations are required to train each gesture recognizer, but convergence often occurs sooner. The entire process requires less than twenty minutes on a 1 GHz Pentium 4, running Linux, with 512 MB of memory.

### 3.2.2 Setting the Confidence Vector

As described in Section 3.1.2, the confidence vector maintains a set of ratings on performance of each pred-

icate recognizer. These ratings range between 0 and 1, and they control the extent to which each predicate recognizer influences the final outcome. In order to determine the ratings in the confidence vector, 20% of the training data is withheld from training. This data is called the "tuning set", and it is used to compute the confidence vector.

As in training, we evaluate each predicate recognizer based on the sensor data and the expected output. However, instead of performing any additional training, we simply record its performance. It is critical that we use data not present in the training set, since this gives an indication of whether the predicate recognizers can generalize to examples outside of the training data. In this way, the confidence vector helps us account for and deal with *overfitting*, a serious problem in many machine learning applications.

Let $\mathbf{a}[i]$ represent the accuracy of the $i^{th}$ predicate recognizer on the tuning set. All predicate recognizers evaluate to either 1 or 0; consequently, if $\mathbf{a}[i] = 0.5$, then the predicate recognizer is only performing at chance. In this case, our confidence rating should be zero, since the predicate recognizer is no better than flipping a coin. On the other hand, if $\mathbf{a}[i] = 1$, then the predicate recognizer is performing perfectly, and a confidence rating of one should be assigned. All of this can be quantified in the following equation:

$$\mathbf{k}[i] = 2(\mathbf{a}[i] - 0.5)^{+} \tag{4}$$

The superscript "+" indicates that the term in parentheses can never go below zero; if it does, it is set to exactly zero, i.e. $(\mathbf{a}[i] - 0.5)^{+} = \max(\mathbf{a}[i] - 0.5, 0)$. A coefficient of two is applied in order to normalize confidence rating to a range between 0 and 1.

# 4 Experimental Results

We have evaluated our approach in the domain of ASL fingerspelling. Specifically, we focus on static signs and use the twenty-four letters which do not contain temporal characteristics – Z and J are omitted. Extending our system to handle spatio-temporal gestures is the subject of future work. Our evaluation proceeds along three dimensions: performance, device-independence, and extensibility. We will show that our approach achieves at least comparable performance to a conventional approach, while providing a level of device-independence and extensibility that are well beyond the capabilities of any known conventional system.

## 4.1 Performance

Performance can be evaluated on two dimensions: speed, and accuracy. Although our system is substantially more complex than many of other approaches to this problem, gesture recognition is still quite fast. Using the same computer hardware described in Section 3.2.1, out system recognizes more than 600 gestures per second. This is faster than the polling rate of our glove device, and we believe that it is certainly fast enough for use in an interactive system.

In order to evaluate the accuracy of our system, we a conventional feedforward neural network as a baseline for comparison. This neural network consumes all twenty-two raw sensor values as input, includes twenty-five hidden nodes, and has twenty-four output nodes. The letter corresponding to the maximally activated output node is considered to be the output. This baseline approach is very similar to the approach used in [13].

Many experiments used the same individual signers in the test set as those who were used to train the system ([6] is an exception). Our testing methodology is substantially more rigorous, because we have attempted to achieve *signer independence*. Specifically, out of sixteen signers in our dataset, twelve were used in training, and four were used in testing. Moreover, we performed only the most cursory calibration, taking less than thirty seconds for each signer. To achieve confidence in our results, we performed ten separate experiments, with different, randomly chosen test and training sets in each. The results reported below are the averages over the ten experiments. In a commercial application, developers would be free to choose a training set that yielded maximum performance, but we constructed our training sets randomly to ensure the integrity of our experiments.

We use only a bigram model for context; this improved accuracy by roughly 10%. We also experimented with a trigram model, but found that it yielded only a marginal improvement beyond the bigram, and required significantly more time to evaluate the system.

Since the baseline approach could not take advantage of context, we compared our approach both with and without context against the baseline. We tested all systems by simulating each user signing every word in the English dictionary. The results are shown in the first line of Table 3. With the help of a bigram context, our system strongly outperformed the baseline. Without the bigram context, our system was slightly better than the baseline. This validates our claim that our approach performs as well as the baseline.

| System | Baseline | Multilayered | Multilayered |
|---|---|---|---|
| Context | no | yes | no |
| CyberGlove 22 | 58.9% | 67.4% | 58.6% |
| CyberGlove 18 | 59.2% | 65.5% | 54.0% |
| DataGlove 16 | 57.8% | 59.4% | 51.2% |
| DataGlove 10 | 45.2% | 40.5% | 32.3% |
| TCAS 8 | 24.1% | 38.6% | 31.5% |
| DataGlove 5 | 20.1% | 25.2% | 8.3% |

**Table 3. Accuracy Results for the Six Gloves with $n = 2$.**

### 4.2 Device Independence

To show that our framework supports device independence, we tested our system on six different glove input devices. The predicate recognizers and confidence vector were retrained for each device; the template matcher and context modules were reused. The first glove, *CyberGlove 22*, is a real glove used in our lab and the other five gloves are real gloves that are simulated by removing sensors from the data files produced by the first glove. All six gloves are described in [17]; their relevant details are as follows. The details of all six gloves are as follows.

1. *CyberGlove 22.* A 22 sensor glove, with a sensor for every joint in the human hand.
2. *CyberGlove 18.* Omits the distal (outermost) joints in each finger. The thumb has no distal joint.
3. *DataGlove 16.* Omits the distal joints, and the wrist flexion and abduction sensors.
4. *DataGlove 10.* Two sensors per finger and thumb. Omits distal joints, wrist flexion and abduction, palm arch, thumb roll, and abduction between fingers.
5. *TCAS Glove 8.* One sensor per finger and thumb, plus thumb roll and palm arch.
6. *DataGlove 5.* One sensor per finger and thumb.

Whereas the baseline neural network had to be entirely retrained for each glove, our system only retrained the low-level gesture recognizers and the confidence vector. Even under these conditions, our system performs at or near the same level as the baseline on nearly every glove.

### 4.3 Extensibility

To demonstrate the extensibility of our system, we conducted the following experiment. We removed letters from the training data when the gesture recognizers were being trained, and then later added their templates to the library. The goal is to determine whether our system could be extended to recognize these new templates, even though they were not in the training set of the gesture recognizers. We successively removed every letter from the training set, one at a time, and then tested the accuracy on that letter. We found that the average accuracy across all twenty-four letters was 92% as good as when the letters were included in the training set. This shows that performance degrades only marginally on new templates that were not part of the training set, and suggests that it is indeed possible to extend the system with new signs and maintain adequate performance.

## 5 Related Work

Glove-based gesture recognition has been explored in a number of studies. In an early study, Murakami and Taguchi [10] used recurrent neural networks to detect signs from Japanese Sign Language. Newby used a "sum of squares" template matching approach to recognizing ASL signs, which is very similar to our template matching component [11]. This system does seem to have met the criterion of extensibility, although this is not mentioned explicitly; the method was chosen because of its fast execution time. More recent studies in gesture recognition has focused on hidden Markov models, which have produced highly accurate systems capable of handling dynamic gestures [6, 8, 14].

The idea of device independence in virtual reality applications has been addressed in a number of studies. One such study, by Faisstnauer et al., describes the Mapper [5], which eases the integration of new devices into virtual reality applications. This study does not tackle the specific issue of gesture recognition, but instead provides a high level software engineering framework for handling heterogeneous devices.

More closely related to our own work is that of Su and Furuta [15]. They propose a "logical hand device" that is in fact a semantic representation of hand posture, similar to our own set of postural predicates. This was done with the express purpose of achieving device independence, but to our knowledge it was never implemented. Our research can be viewed as an implementation and experimental validation of these ideas.

## 6 Conclusion and Future Research

The experimental results described here are preliminary evidence that our system does indeed achieve our two goals of device independence and extensibility. We would like to make further experiments along these lines. Regarding the claim of device independence, we

would like to test our system with other real gloves, rather than merely simulating them. As for the extensibility claim, we would like to show that our predicate recognizers, trained on ASL data, can actually support entirely different sign languages, such as Japanese or Russian Sign Language. Ultimately, we would like to go even further, and show that we can support applications outside of sign language altogether, such as virtual reality.

Since we have focused only on postural predicates, our predicate recognizers use feedforward neural networks, which we believe are well-suited to this task. We hope that the same basic framework can be applied to temporal predicates, perhaps using recurrent neural networks or hidden Markov models for the predicate recognizers. The questions of how to build these temporal predicate recognizers, and how to incorporate them into our existing framework are clearly crucial next steps towards creating a useful system.

**Acknowledgements**

We thank Yong Zeng for preparing the figures, and for his assistance with the experimental evaluation of the system.

## References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[2] J. Eisenstein, S. Ghandeharizadeh, L. Golubchik, C. Shahabi, D. Yan, and R. Zimmermann. Multi-layer gesture recognition: An experimental evaluation. Technical Report 02-766, University of Southern California Department of Computer Science, 2002.

[3] J. Eisenstein, S. Ghandeharizadeh, L. Huang, C. Shahabi, G. Shanbhag, and R. Zimmermann. Analysis of clustering techniques to detect hand signs. In *Proceedings of the 2001 International Symposium on Intelligent Mulimedia, Video and Speech Processing*, Hong Kong, May 2001.

[4] J. Eisenstein, S. Ghandeharizadeh, C. Shahabi, G. Shanbhag, and R. Zimmermann. Alternative representations and abstractions for moving sensors databases. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM)*, Atlanta, GA, November 5-10, 2001.

[5] C. Faisstnauer, D. Schmalstieg, and Z. Szalavári. Device-independent navigation and interaction in virtual environments. In *Proceedings of the VRST Adjunct Workshop on Computer Graphics*, Taipei, Taiwan, November 5-6, 1998.

[6] G. Fang, W. Gao, X. Chen, C. Wang, and J. Ma. Signer-independent continuous sign language recognition based on SRN/HMM. In *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time*, pages 90–95, Vancouver, BC, Canada, July 2001.

[7] Immersion Corporation, San Jose, CA. *CyberGlove Reference Manual*, 1998.

[8] R.-H. Liang and M. Ouhyoung. A sign language recognition system using hidden markov model and context sensitive search. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'96)*, pages 59–66, Hong Kong, June 1996.

[9] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 2000.

[10] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of the Conference on Human Factors and Computing Systems (CHI'91)*, pages 237–242, New Orleans, Louisiana, 1991.

[11] G. B. Newby. Gesture recognition based upon statistical similarity. *Presence*, 3(3):236–243, 1994.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:318–362, 1986.

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

[14] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, Zürich, 1995.

[15] S. A. Su and R. Furuta. A logical hand device in virtual environments. In *Proceedings of the ACM Conference on Virtual Reality Software and Technology (VRST'94)*, pages 33–42, Singapore, August 23-26, 1994.

[16] C. Y. Suen. N-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):164–172, April 1979.

[17] C. Youngblut, R. E. Johnson, S. H. Nash, R. A. Wienclaw, and C. A. Will. Review of virtual environment interface technology. Technical Report IDA Paper P-3186, Log: H96-001239, Institute for Defense Analysis, 1996.