Activity Analysis of Sign Language Video

Neva Cherniavsky

September 19, 2007

Abstract

The MobileASL project aims to enable Deaf people to communicate over video cell phones in their native language, American Sign Language (ASL). Real-time video over cell phones can be a computationally intensive task that quickly drains the battery, rendering the cell phone useless. In previous work we have shown that by recognizing activity in the video, we can drop the frame rate during unimportant segments without significantly harming intelligibility, thus reducing the computational burden. Here, we review current research in the related area of sign language recognition. We then propose to expand our analysis of sign language video to include recognizing finger spelling, and ultimately, to sign segmentation.

1 Introduction

Mobile phones are rapidly becoming ubiquitous, with over 2.68 billion mobile phone subscribers worldwide [28]. In the United States, federal law has long mandated accessibility to the telephone network for the Deaf through subsidized telephone typewriters (TTY) and, more recently, video relay service. However, there is no equivalent service for access to mobile telecommunications. While mobile devices such as Blackberries and Sidekicks are used frequently within the signing Deaf community, they require communication through email or instant messaging, both of which are slower than signing and necessitate the use of a different language.

The MobileASL project [8, 12] aims to expand accessibility for Deaf people by compressing sign language video to enable mobile phone communication. We envision users taking and receiving video on a typical mobile phone. With today's bandwidth limitations [22] and the low processing power available on phones, real-time sign language communication is not feasible using current video compression technology. However, in the same way that there are characteristics unique to speech that allow speech to be compressed more than standard audio, sign language has distinct features that should enable better compression than is typical for video. For example, much of the grammar of sign language is found in the face. Higher intelligibility at lower bit rates can be realized by allocating more bits to the face and less to the rest of the image [8].

In this work, we focus on recognizing activity in sign language video in order to make adjustments that might increase or maintain intelligibility while decreasing cost. Cost can be measured in several ways: dollar value, if users are expected to pay based on how much data they transmit; processing power, so that real-time compression may be achieved on a standard phone; and battery life, since a short-lived phone is not very useful. In previous work [11], we show that lowering the frame rate on the basis of the activity in the video could lead to savings in data transmitted and processor cycles, and thus power, while not too negatively affecting intelligibility. Frame rates as low as 6 frames per second can be intelligible for signing, but higher frame rates are needed for fingerspelling [19, 47, 30]. Because conversation involves turn-taking (times when one person is signing while the other is not), we save power as well as bit rate by lowering the frame rate during times of not signing, or "just listening" (see Figure 1). We focus on some simple features of sign language activity and use a support vector machine to demonstrate proof-of-concept that signing can be distinguished from not signing in real time.

Here, we further examine the features that may be reasonably extracted in real time and the types of machine learning techniques that can be applied to the features. We review our previous work and propose new directions for future research. Specifically, we explore applying new feature extraction and machine learning techniques to

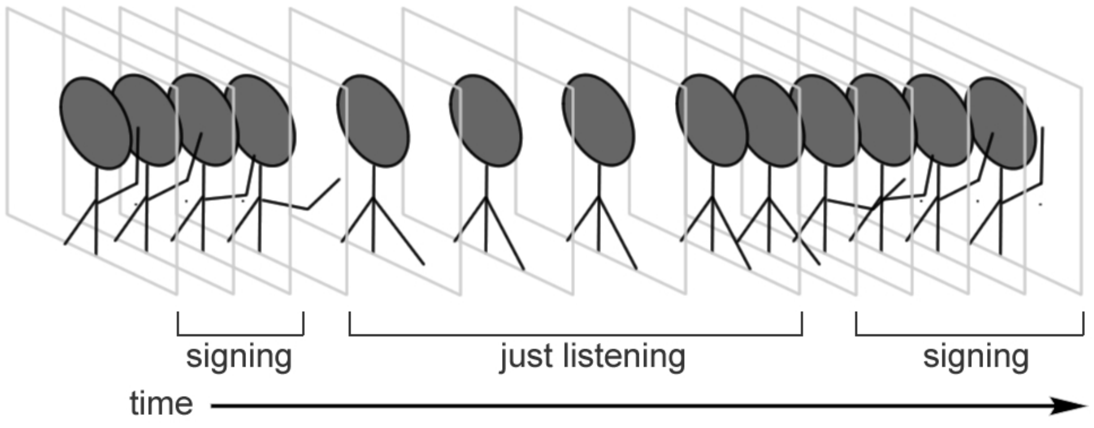

Figure 1: From left to right: a sufficient video frame rate is chosen when the signer is signing, the frame rate decreases when the signer is not signing (or just listening), and increases again when the signer begins signing.

recognizing finger spelling, in order to increase intelligibility. We then consider other types of activity during which we might be able to lower the frame rate. In general we would like to be able to drop the frame rate in such a way that we only send the most important frames, thus maximizing intelligibility at the minimum cost. To this end, we aim to recognize *movement epenthesis*, which is a transition between sequences of signs that in itself conveys no meaning. Recognizing movement epenthesis is equivalent to sign segmentation, and thus could also be the first step in a continuous sign language recognition engine.

The paper is organized as follows. In the next section, we discuss research in the related area of sign language recognition. We delve into the details of the methods that we plan on applying in our future research. In section 3 we discuss the research completed thus far in detecting signing and not signing frames from the video. Section 4 contains our proposal for the two directions for future research: recognizing finger spelling to increase intelligibility, and recognizing movement epenthesis. We detail the steps toward completing our proposed research with a timeline. We conclude in section 5.

2 Background

Sign language recognition is well-studied in the literature [39]. The problem of recognizing structured, three-dimensional gestures is quite difficult and progress has been slow; the state-of-the-art in sign language recognition is far behind that of speech recognition, with limited vocabularies, signer dependence, and constraints on the signers. Solutions must combine techniques from computer vision, natural language processing, and machine learning.

Signs in ASL are made up of five parameters: handshape, movement, location, orientation, and nonmanual signals [53]. Recognizing sign language is mostly constrained to recognizing the first four. Nonmanual signals, such as the raising of eyebrows (which can change a statement into a question) or the puffing out of cheeks (which would add the adjective "big" or "fat" to the sign) are usually ignored in the literature. Without nonmanual signals, any kind of semantic understanding of sign language is far off. Nonetheless, progress has been made in recognition of manual signs.

There are two main parts to sign language recognition, both of which are applicable to our task. The first is feature extraction, that is, selecting the most relevant features from the data for training and testing. The second is machine learning - choosing a technique from the many available for training and testing on the feature data in order to obtain the best results. Both steps have tradeoffs in terms of robustness, efficiency, and accuracy. Below we review some common methods and their advantages and disadvantages. We then discuss the current research on sign segmentation.

2.1 Feature extraction for sign recognition

The most effective techniques for sign language recognition use direct-measure devices such as data gloves to input precise measurements on the hands. These measurements (finger flexion, hand location, roll, e.g.) are then used as the features for training and testing purposes. While data gloves make sign recognition an easier problem to solve, they are expensive and cumbersome, and thus only suitable for constrained tasks such as data input at a terminal kiosk [1]. We focus instead on vision-based feature extraction.

The goal of feature extraction is to find a reduced representation of the data that models the most salient properties of the raw signal. Following Stokoe's notation [49], manual signals in ASL consist of handshape, or *dez*; movement, or *sig*; location, or *tab*; and palm orientation, or *ori*. Most feature extraction techniques aim to recognize one or more of these parameters. By far the most common goal is to recognize handshape. Some methods rotate and reorient the image of the hand, throwing away palm orientation information [36]. Others aim only to recognize the handshape and don't bother with general sign recognition [26, 25, 36]. Location information, or where the sign occurs in reference to the rest of the body, is the second most commonly extracted feature. Most methods give only partial location information, such as relative distances between the hands or between the hands and the face. Movement is sometimes explicitly extracted as a feature, and other times implicitly represented in the machine learning portion of the recognition. Palm orientation is not usually extracted as a separate feature, but comes along with hand shape recognition.

Table 1 summarizes the feature extraction methods of the main works on sign language recognition. The last column gives the time complexity of the technique. If feature extraction is too slow to support a frame rate of 5 frames per second (fps), it is not real-time and thus not suitable to our purposes. This includes Huang et al. and Chen et al.'s Fourier descriptors to model hand shape [10, 25]; Cui and Weng's pixel intensity vector [14]; Huang and Jeng's active shape models [26]; and Tamura and Kawasaki's localization of the hands with respect to the body [51]. Though the time complexity was unreported, it is likely that Imagawa et al.'s principal component analysis of segmented hand images is not real-time [27]. Yang et al. also did not report on their time complexity, but their extraction of motion trajectories from successive frames uses multiple passes over the images to segment regions and thus is probably not real-time [60]. Nonetheless, it is interesting that they obtain good results on isolated sign recognition using only motion information.

Bowden et al. began by considering the linguistic aspects of British sign language,

Features	**Part of sign**	**Constraints**	**Time**	**1st Author**
Real-time (measured in frames per second)				
COG; contour; movement; shape	dez, tab, sig	Isolated signs	25 fps	Bowden [7]
COG	dez, ori	Colored gloves; uniform background; isolated signs	13 fps	Assan [2] Bauer [3]
COG, bounding ellipse	dez, tab, ori	Colored gloves; uniform background; no hand-face overlap; strong grammar	10 fps	Starner [48]
COG	dez, tab	One handed isolated signs	?	Kobayashi [33]
COG; Area; No. protusions; motion direction	dez, tab, sig, ori	Uniform background; isolated signs	?	Tanibata [52]
Not real-time (measured in seconds)				
Fourier descriptors; optical flow	dez, sig	Hand constantly moving; one handed isolated signs	1 s	Chen [10]
COG	dez, tab	Uniform background; one handed isolated signs	3 s	Tamura [51]
Fourier descriptors	dez	Hand constantly moving; dark clothes; uniform background; hand shape only	10 s	Huang [25]
Active shape models	dez	Uniform background; hand shape only	25 s	Huang [26]
Intensity vector	dez	Hand constantly moving; one handed isolated signs; away from face	58.3 s	Cui [14]
PCA	dez	Isolated signs classified by begin and end hand shape	?	Imagawa [27]
Motion trajectory	sig	Isolated signs	?	Yang [60]

Table 1: Summary of feature extraction techniques and their constraints. COG: center of gravity of the hand. *dez*: hand shape, *tab*: location, *sig*: movement, *ori*: palm orientation.

and made this explicitly their feature vector [7]. Instead of orientation, British sign language is characterized by the position of hands relative to each other (*ha*). They recognize ha via COG, tab by having a two dimensional contour track the body, sig by using the approximate size of the hand as a threshold, and dez by classifying the hand shape into one of six shapes. They use a rules-based classifier to group each sign along the four dimensions. Since they only have six categories for hand shape, the results aren't impressive, but the method deserves further exploration.

Most promising for our purposes are the techniques that use the center of gravity (COG) of the hand and/or face. When combined with relative distance to the fingers or face, COG gives a rough estimate about the hand shape, and can give detailed location information. One way to easily pick out the hands from the video is to require the subjects to wear colored gloves. Assan and Grobel [2] and Bauer and Kraiss [3] use gloves with different colors for each finger, to make features easy to distinguish. They calculate the location of the hands and the COG for each finger, and use the distances between the COGs plus the angles of the fingers as their features. Tanibata et al. use skin detection to find the hands, then calculate the COG of the hand region relative to face, the area of hand region, the number of protusions (i.e. fingers), and the direction of hand motion [52]. Signers were required to start in an initial pose. Kobayashi and Haruyama extract the head and the right hand using skin detection and use the relative distance between the two as their feature [33] . They recognized only one-handed isolated signs. Starner et al. use solid colored gloves to track the hands and require no hand-face overlap and a strong grammar [48]. Using COG plus the bounding ellipse of the hand, they obtain hand shape, location, and orientation information.

2.2 Machine learning for sign recognition

Many of the researchers in sign language recognition use neural networks to train and test their systems [16, 17, 21, 25, 37, 54, 58, 60]. Neural networks are quite popular since they are simple to implement and can solve some complicated problems well. However, they are computationally expensive to train and test; they require many training examples lest they overfit; and they give a "black-box" solution to the classification problem, which does not help in identifying salient features for further refinement [44].

Decision trees and rules-based classifiers are another major way that researchers recognize sign language [43, 23, 27, 31, 45, 51]. These are quite fast, but sensitive to the rules chosen. Some works incorporate decision trees into a larger system that contains some other, more powerful machine learning technique [38]. That idea holds promise; for instance, it makes sense to divide signs into two-handed and one-handed using some threshold, and then apply a more robust shape recognition algorithm.

We focus on methods that are fast (real-time) and perform well for sign recognition. We plan on using both hidden Markov models and boosting in our proposed research. We also would like more robust tracking techniques to obtain better features, and so explore those at the end of this section.

2.2.1 Hidden Markov models

The majority of research in sign language recognition uses hidden Markov models for sign classification [2, 3, 10, 17, 21, 26, 48, 52, 57, 59, 62]. Hidden Markov models are appealing because they have been successfully applied to speech recognition.

A Markov model is simply a finite state machine in which the next state depends only on the current state, that is:

$$Pr(X_{t+1} = x | X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_1 = x_1) = Pr(X_{t+1} = x | X_t = x_t).$$

In a regular Markov model, the state is known; in a hidden Markov model, the state is not known, but there is an observation at each step and some probability that each state produced the observation. Intuitively, suppose there are two friends, Bob and Alice. Every day Alice calls Bob and tells him what she did that day. Alice either goes shopping, goes to a movie, or plays baseball, depending on the weather (see Figure 2). So, if it's sunny, she's likely to play baseball, and if it's rainy, she'll probably go to a movie. Additionally, when it's sunny one day, there's a good chance

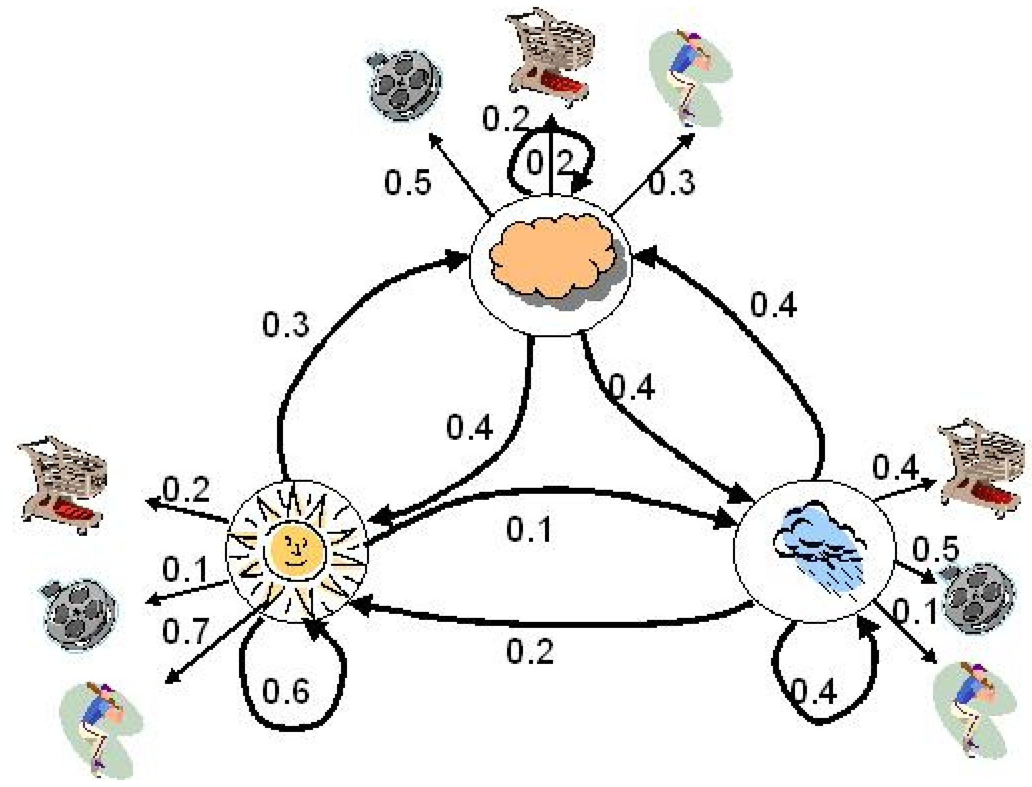

Figure 2: Graphical representation of a hidden Markov model. The hidden states correspond to the weather: sunny, cloudy, and rainy. The observations are Alice's activities.

it will be sunny the next day; but if it's cloudy, it could rain the next day. For Bob, the weather represents the hidden states, and based on Alice's activities over a series of days, he can guess what the weather has been.

Formally, a hidden Markov model is defined by the tuple $\lambda = (N, M, A, B, \pi)$:

- N is the number of states, which can be arbitrary but usually conforms to some underlying physical property; the individual states are denoted $S = \{S_1, \ldots, S_N\}$ and the state at time t is q_t.
- M is the number of distinct observation symbols; the individual symbols are denoted $\{v_1, \ldots, v_M\}$.
- $A = \{a_{ij}\}$ is the state transition probability, where $a_{ij} = Pr(q_{t+1} = S_j | q_t = S_i)$, $1 \leq i, j \leq N$.

- $B = \{b_j(k)\}$ is the observation symbol probability, where $b_j(k) = Pr(v_k \text{ at } t | q_t = S_j)$, $1 \leq j \leq N$, $1 \leq k \leq M$.
- $\pi = \{\pi_i\}$ is the initial state distribution, where $\pi_i = Pr(q_1 = S_i)$, $1 \leq i \leq N$.

There are three canonical problems for hidden Markov models [41]. First, given an observation sequence and a model, find the probability that the model produced the observation sequence. Second, given an observation sequence and a model, find the corresponding state sequence that best explains the observation sequence. Third, given an observation sequence, adjust the parameters of the model to maximize the probability of the observation sequence given the model.

The first canonical problem is the one used to solve speech recognition. In speech recognition, a model is created for each phoneme. These are tied together into models representing words. Then, an utterance is the observation and the word model that most likely produced that utterance is the recognized word. Vogler uses this method for sign recognition [57].

The solution for this problem is the forward-backward algorithm [4]. Define the forward variable as $\alpha_t(i) = Pr(O_1 O_2 \cdots O_y, q_t = S_i | \lambda)$. The forward variable is the probability of seeing the observation sequence up to time t and being in state i at time t. This can be solved for inductively:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(O_{t+1})$$

That is, sum over all the possible ways of arriving at state j having seen the current observation sequence, and multiply times the probability of observing O_{t+1} at state j. Then the probability of the sequence given the model is $Pr(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$. The backward variable is defined analogously to the forward variable, and is included here because it will be useful later: $\beta_t(i) = Pr(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda)$.

Starner et al. use the second canonical problem for sign recognition [48]. They apply a hidden Markov model for continuous sign recognition, so that the model implicitly segments the signs. The model is sensitive to transitions between signs, because the starting point of a sign will change depending on the previous sign. They mitigate this by imposing a strong grammar, that is, only allowing certain signs to come before or after each other.

Finding the best state sequence is usually solved with the Viterbi algorithm [56, 18]. Define the best score along a single path that ends in state i as

$$\delta_t(i) = \max_{q_1 \cdots q_{t-1}} Pr(q_1 \ldots q_t = i, O_1 \cdots O_t | \lambda).$$

This can be solved via dynamic programming, as $\delta_{t+1}(j) = (\max_i \delta_t(i) a_{ij})\, b_j(O_{t+1})$. The state sequence is retrieved by saving the values that maximized $\delta_t(i)$ at each time step.

The last canonical problem is equivalent to training the models. This is by far the hardest problem to solve. There is no known way to optimally find the model that maximizes the probability of the observation sequence [41]. Instead, researchers apply iterative methods that search for a local maximum. A common solution is the Baum-Welch algorithm, also known as expectation-modification (EM) [15]. Essentially, the algorithm works by calculating the forward and backward probability for each state, as defined above. Then the model is updated by setting $\overline{\pi}$ to the expected number of times in state S_i at time 1; $\overline{a_{ij}}$ to the expected number of transitions from S_i to S_j divided by the expected number of transitions from state i; and $\overline{b_j(k)}$ to the expected number of times in state j and observing v_k divided by the expected number of times in state j. Details are available in Rabiner's tutorial [41].

2.2.2 Boosting

Boosting is a general technique for creating one good classifier out of many marginal ones. Intuitively, suppose we wanted to classify email into spam or not spam. We could use a rule of thumb such as "if this email contains the word Viagra, it is spam". This rule would not work very well, but would work better than just random guessing. Boosting works by taking many such rules, called *weak learners* and iteratively weighting them based on their classification. The combination of these rules is a *strong learner* that classifies the data well.

```
**AdaBoost**$(< (x_1, y_1), \ldots, (x_m, y_m) >,\ x_i \in X,\ y_i \in \{-1, +1\},\ h)$
    1. Initialize weighting $D_i(1)$ to $1/m$, $\forall 1 \leq i \leq m$
    2. For $t = 1, \ldots, T$ :
        (a) Train $h$ using weights $D_t$
        (b) Resulting weak hypothesis is $h_t : X \rightarrow \{-1, +1\}$ with error $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
        (c) $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
        (d) $D_{t+1}(i) = \begin{cases} D_t(i)e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ D_t(i)e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$
                        $= D_t(i) \exp(-\alpha_t y_i h_t(x_i))$
        (e) Normalize $D_{t+1}$ so that it is a distribution.
    3. Return the final hypothesis: $H(x) = \text{sign}\left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$.
```

Figure 3: The AdaBoost algorithm [46]. X is the feature set, h is the weak learner.

A popular boosting algorithm, due to its practicality, adaptivity, and resistance to overfitting, is AdaBoost [20]. AdaBoost is quite simple to implement and can be described in a few lines of pseudocode (Figure 3). When examples are misclassified, it adapts by weighting those higher and forcing the weak learner to improve. Furthermore, Freund and Schapire [20] proved that if each weak learner is slightly better than random, then the training error drops exponentially fast. Thus, given enough training data, AdaBoost can generate a classifier with an arbitrarily low error rate.

Viola and Jones built a system on top of AdaBoost to detect faces and other objects in real-time (15 fps) [55], which was then used by Ong and Bowden [38] for hand shape recognition. Viola and Jones start with simple image features based on the sum of image pixels within adjacent rectangles. The two-rectangle feature is the difference between the sum of two adjacent rectangles; the three-rectangle feature is the sum of a center rectangle subtracted by the two outside rectangles; and the four rectangle feature is the difference between diagonal rectangles. Since they are essentially taking all rectangles starting at a scale of 24×24, the number of features is enormous. To calculate them quickly, they create an integral image I' that contains the row and column sum of the original image I: $I'(x, y) = \sum_{i \leq x, j \leq y} I(i, j)$.

It would be impossible to detect faces in real time using all of the rectangular features available. Instead, Viola and Jones use AdaBoost to both train the classifier and select which features are most useful. The weak learning algorithm is constrained to use a single feature, so at each time step in the AdaBoost algorithm, the training examples are classified only by one rectangular feature (that has some error rate lower than 50%). Then the number of iterations of the algorithm is the same as the number of features used to classify the data. The authors also quickly reject subwindows that can't possibly contain the face by using a cascade of classifiers. The early, simplest classifiers have high false positive rates but no false negatives; the later classifiers have a tougher job to do but operate over a much smaller data set.

Ong and Bowden extend this idea to template matching for hand shape recognition [38]. In template matching, training examples are clustered to create iconic images of the particular hand shapes. For testing, the closest matching iconic image is chosen. They create a tree structure of classifiers instead of just a cascade, in which each layer recognizes more specific characteristics. For training, the hand images are clustered into different groups with histograms based on a log-polar coordinate system. Their error rate on hand shape detection was 2.6%.

In similar work, Lockton and Fitzgibbon use template matching of hands that they have segmented using special lighting conditions and a wrist band [36]. These images, which are well-aligned, are then clustered into iconic images. Then, instead of matching the image via nearest neighbor (as is usual in template matching), they create a weak classifier based on just one pixel. They build on top of that with another pixel, boosting at each level, so that in the end they query 30-fold fewer pixels than the nearest neighbor algorithm. Their success rate is extremely high, with 99.87% test hand shapes recognized.

2.2.3 Tracking

As noted above, we would like to use the centers of gravity of the hands and face as a feature, and tracking provides an accurate estimation. AdaBoost can be used to track hands, but the most common way to do so is via a Kalman filter [27, 24], which is sometimes used in conjunction with other color-based algorithms [63, 6].

The Kalman filter is an algorithm for estimating the next state of a linear system in the presence of noise [32]. Kalman filters work by repeatedly estimating the process state at a point in time, measuring, and adjusting the estimation parameters. An optimal Kalman filter minimizes the mean squared error between the estimated state and the measured state. Over time, if there is a large variance in the signal, the Kalman filter will put less weight on the measurements and more weight on its previous prediction; conversely, if the variance is small, the filter will weight the measurements more heavily.

Binh et al. [5] use a combination of pseudo 2-D hidden Markov models and a Kalman filter to track hands. Feedback from the pseudo 2D HMM is used to train the Kalman filter and the results from the Kalman filter are used as input to the pseudo 2D HMM. The authors use the centroid of the hand, the velocity, and the width and height of the bounding box as their feature vector for the Kalman filter. The measurements are obtained by tracing the state sequence of the pseudo 2-D hidden Markov model. In turn, the Kalman filter predicts where the hand will be in the next frame, and that is the image sent to the pseudo 2D hidden Markov model.

Pseudo 2-D hidden Markov models have been used successfully in optical character recognition [34]. A pseudo 2D hidden Markov model is a restricted version of a two dimensional hidden Markov model. Two dimensions are desireable because they lead to greater recognition of two dimensional images; however, recognizing a fully connected 2-D HMM is an NP-complete problem. A pseudo 2-D HMM can be solved in polynomial time, but is general enough to lead to better recognition of two dimensional data. The traditional restriction is on the transition probabilities between states. Only three types of transitions are allowed: a self-transition, a transition to one state ahead, and a transition to two states ahead. This restriction holds for both the 1-D HMMs that make up superstates, and between the superstates themselves. Pseudo 2-D HMMs hold some promise for shape recognition.

Closely related to the Kalman filter is another probabilistic algorithm, Isard and Blake's CONDENSATION tracker [29]. CONDENSATION differs from the Kalman filter in that it is able to represent a wider range of probability distributions. It is essentially a particle filter. As the input data evolves, indicating a changing probability distribution, CONDENSATION is able to keep up by randomly sampling at each step to create a new distribution. CONDENSATION is most useful for tracking multiple objects in a complex scene; the method bears further exploration, but we will focus on using the simpler Kalman filter to start.

2.3 Sign segmentation

Our ultimate goal in our proposed research is to recognize movement epenthesis, which is equivalent to sign segmentation. Some researchers have attempted to solve this problem as a part of a larger continuous sign recognition engine.

Sagawa and Takeuchi use motion cues and finger flexion data (from a direct measure device) to estimate when one sign ends and another begins [45]. If there is a large change in motion trajectory and the glove data indicates low motion, the frame is considered a boundary candidate. This method results in an error rate of 19.8%, possibly because signs do not always end/begin with a change in trajectory. Liang and Ouhyoung similarly use a threshold on finger flexure, in this case classifying frames with too much finger motion as transitions [35]. Their method suffers from the same problem as the first; namely, the rule is too rigid to properly classify all types of transitions (and may mistake signs for transitions). Fang et al. [17] uses an approach close to what we wish to implement. Operating on direct measure device data, they use a machine learning technique (in their case, a recurrent neural network) to classify frames as left boundary, right boundary, or interior of segment. For recognition, they send the segmented signs to a hidden Markov model. Recognition accuracy was not impressive, but segmentation accuracy was 98.8%.

Movement epenthesis has some similarity to coarticulation effects in speech recognition. The common solution for coarticulation is to use context dependent models, that is, chain together phonemes into biphone or triphone models [21]. Vogler argues convincingly that this method does not work well on sign language, because the size of the search space is increased substantially when signs are chained together [57].

He instead suggests explicitly modeling movement epenthesis, which is the approach we will take. He uses hidden Markov models representing each unique ending sign position to starting sign position. He shows his method works better than either context independent modeling or biphone models.

Yang et al. recently attacked the problem of sign segmentation using conditional random fields [61]. The features are represented in a histogram of the displacement of points along a three-frame motion trajectory. They first choose "key frames", which are those most different from one another and from frames labeled signing. They then use a conditional random field (a generalization of hidden Markov models that are undirected and relax constraints on transition probabilities) to select which key frames are sign boundaries. They report 85% sign segmentation accuracy, versus 60% using hidden Markov models. Since their method requires knowing the entire data sequence in advance, it is not real time.

3 Completed Research

The first type of activity we have attempted to recognize from sign language video is simply signing or not signing [11]. We show that cost savings may be achieved by dropping the frame rate in times of not signing. However, dropping the frame rate is not feasible if it results in low use of the phones because the signing is unintelligible. To determine the effect of dropping the frame rate on user experience, we conducted user studies. Those studies justified our approach for cost savings; the users reported that the experience was not too irritating and that they could still understand the other side of the conversation. We then focus on some simple features of sign language activity and use a support vector machine to demonstrate proof-of-concept that signing can be distinguished from not signing in real time. Since publication, we have applied hidden Markov models to the same features, with similar results.

3.1 Features

The H.264 video encoder that we use for MobileASL has motion information in the form of motion vectors. For a video encoded at a reasonable frame rate, there is not much change from one frame to the next. H.264 takes advantage of this fact by first sending all the pixel information in one frame, and from then on sending a vector that corresponds to the part of the previous frame that looks most like this frame plus some residual information. More concretely, each frame is divided into macroblocks that are 16×16 pixels. The compression algorithm examines the following choices for each macroblock and chooses the cheapest (in bits) that is of reasonable quality:

1. Send a "skip" block, indicating that this macroblock is exactly the same as the previous frame.
2. Send a vector pointing to the location in the previous frame that looks most like this macroblock, plus residual information.
3. Subdivide the macroblock and reexamine these choices.
4. Send an "I" block, essentially the macroblock uncompressed.

Choices 2 and 3 have motion vectors associated with them; choice 4 does not. Choice 1 means no motion at all; choice 2 might indicate a big, sweeping motion, while choice 3 might indicate small, rapid movements. Choice 4 usually indicates the most motion of all, since the encoder only resorts to it when it cannot find a section of the previous frame that matches this macroblock. Figure 4 shows a visualization of the macroblocks, with the subdivisions and motion vectors. The subdivisions around the signer's right hand indicate small, quick movements, while the left arm and the face are exhibiting slower, broader motions.

For each frame, we can obtain either motion vector information for each macroblock or an indication that the encoder gave up. This is quite useful for determining what kind of activity is taking place in the video. If we know the hands are involved in big motions, we can classify the frame as a signing frame; conversely, if the hands and

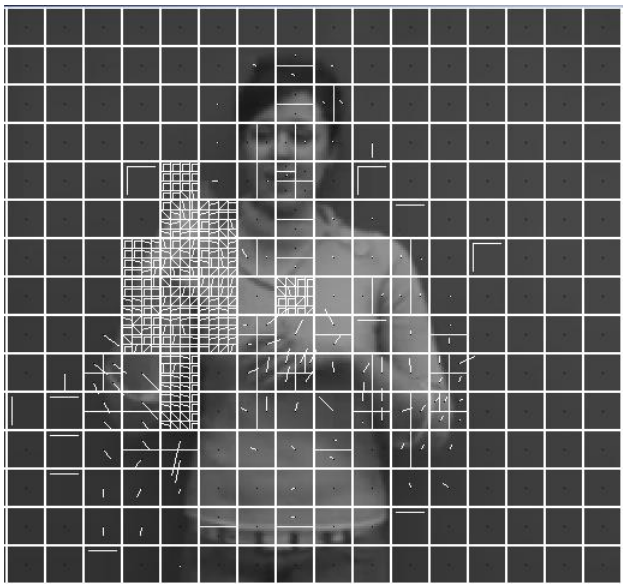
Figure 4: Visualization of the macroblocks. The lines emanating from the centers of the squares are motion vectors.

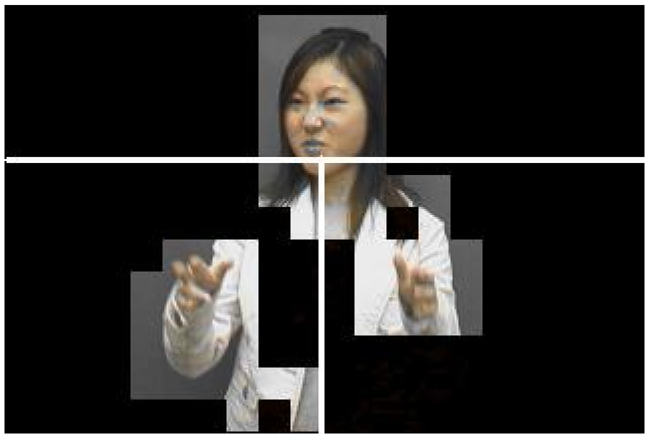
Figure 5: Macroblocks labeled as skin and the corresponding frame division.

face are not moving very much, we can classify the frame as not signing. Thus, for each frame, we will obtain features from the encoder that will help us classify it.

We do not need all of the motion vector information from all of the macroblocks. Instead, we would prefer to focus on the face and hands. We perform skin-detection on the video to determine the macroblocks most likely to contain the face and hands. The skin detection is based on the color value of the pixels, and so can be performed in real-time. We then divide the frame into three parts: the top third, corresponding to the face, and the bottom two thirds divided in half, corresponding to the left and

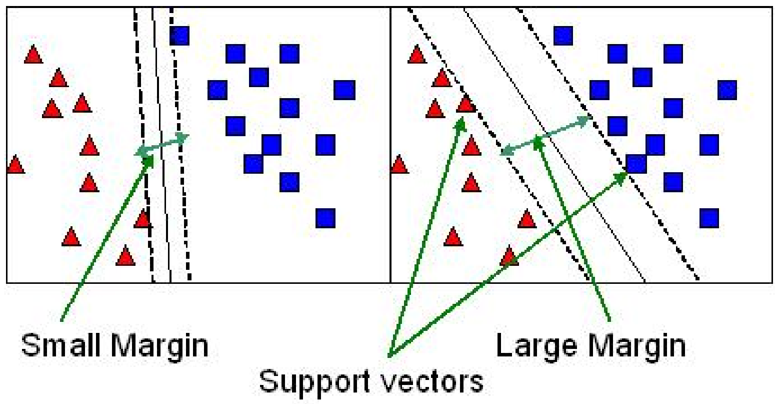

Figure 6: Optimal separating hyperplane.

right hands. Any macroblock with majority skin pixels we classify as skin. For those macroblocks, we calculate a summary motion vector for the face, right hand, and left hand. As an additional feature, we count the overall number of I-blocks in the frame. Figure 5 shows the macroblocks classified as skin and the frame division. Note that this simple method won't always correspond to the face and hands, and yet yields reasonable results. We can expect that by augmenting and improving the features, we will improve the classifier.

3.2 Support vector machines

A well-known solution to the classification problem is Support Vector Machines (SVM) [13]. A support vector machine is an algorithm that, given labeled training data in the form of features and their classes, determines the optimal separating hyperplane that maximizes the margin (see Figure 6). The hyperplane is not necessarily in the same dimension as the feature space; in fact, it is usually transformed non-linearly to a higher dimensional space in which greater separation may be achieved. SVMs are in the group of linear classifiers, which also includes AdaBoost. In fact, the two are related in that they both work by optimizing the margin, as detailed by Raetsch et al. [42].

We use libsvm [9], a publicly available software package, to train and test our data. The kernel function we apply is the standard radial basis function. We test on four conversational videos. In each video, the same signer "Gina" is filmed by a stationary camera, and she is signing roughly half of the time. The background is uniform and she wears long sleeves. We train on three of the videos and test on the fourth.

Since submitting our paper, we have improved the results for the SVM by also considering the classification of frames immediately previous to the one to be classified. We look at the classification returned by the SVM for the two frames before this one, plus the current classification, and output the majority vote. We also try a four frame and five frame window. We experimented with different weightings on each frame, but found weighting them equally worked best. Table 2 contains our results.

The SVM returns a classification based on which side of the hyperplane the test feature vector is on. Furthermore, it also returns the distance between the hyperplane and the feature vector. The distance can be viewed as a confidence value. If a feature vector is far from the dividing hyperplane, we are very certain of its classification. On the other hand, if a feature vector is close to the hyperplane, we are unsure if the classification is correct. In the future we plan to use this additional information to combine SVM with another machine learning technique, such as AdaBoost.

3.3 Hidden Markov models

There are two main ways to use hidden Markov models to classify our data. The first is to train two models, one for signing and one for not signing, and then recognize the test data by selecting the model that maximizes the probability of the observations. This is the approach used in speech recognition and by Vogler [57] for sign recognition. The second is to attempt to represent the entire video sequence with one model, and then recover the state sequence; presumably, the states will correspond to "signing" and "not signing". Starner et al. [48] used this technique for general sign recognition, but imposed a strong grammar so that their underlying model would be accurate and they would be able to recognize the signs by knowing the state. For our purposes, it is unclear how to model signing and not signing except with two states, and this is too few for the HMM to work well. Thus we choose to follow the first approach, and leave the second to future work.

Test video	SVM	SVM-3	SVM-4	SVM-5	HMM-3	HMM-4	HMM-5
gina1	87.8%	**88.8%**	87.9 %	88.7 %	87.3%	88.4%	88.4%
gina2	85.2%	87.4%	**90.3%**	88.3%	85.4%	86.0%	86.8%
gina3	90.6%	**91.3%**	91.1%	**91.3%**	87.3%	88.6%	89.2%
gina4	86.6%	87.1%	**87.6 %**	**87.6%**	82.6%	82.5%	81.4%

Table 2: Recognition results for the sliding window SVM and HMM. The number next to the method indicates the window size. The best results for each video are in bold.

Our feature data must be more than one frame, to capture the temporal nature of the problem. If we only have one observation per "word", the model degenerates into a single instance of flipping a biased coin, or having some probability that the frame is signing or not signing, and doesn't harness any of the useful information about what came before the observation. We start with a three frame sliding window. Our feature vector x is thus 3×7, where each column consists of the motion vector components used in the SVM: (face$_x$, face$_y$, right$_x$, right$_y$, left$_x$, left$_y$, #I blocks). The first column is the components of the frame two before this one, the second column is the components of the frame immediately previous to this one, and the last column is the components of the current frame. We then try our technique on a four frame and five frame sliding window. Two three-state hidden Markov models are trained on labeled feature data. The first is trained only on signing examples, and the second is trained only on not signing examples. The widely used, publicly available HTK package [50] is used for the hidden Markov model implementation.

Our results are in Table 2. It is interesting to note that the recognition accuracy increased for the most part as the sliding window size increased. The one exception was on "gina4", which had poor results overall for the HMM. This merits further investigation.

4 Proposed Research

We propose to further our work in activity analysis of sign language video. We would like to recognize finger spelling frames, in order to adjust the frame rate as appropriate. More generally, we want to drop frames that are unimportant and send frames that are. To this end, we aim to recognize movement epenthesis, the transition between signs that in itself conveys no meaning. Our hypothesis is that we can recognize activity, specifically finger spelling and movement epenthesis, from sign language video without constraints on the signers, in real time using known algorithms for feature extraction and machine learning.

4.1 Detecting signing, not signing, fingerspelling

We will further our work on detecting signing and not signing by adding improved features, trying new methods, and attempting to recognize finger spelling. So far we have used a support vector machine and hidden Markov models to recognize signing and not signing. Our work in hidden Markov models uses a separate model for each class. We will augment our research by trying to classify via the state sequence version of hidden Markov models.

We will improve our features via better tracking of the hands and face. We will implement a Kalman filter and use color cues to accurately determine the centers of gravity of the hands and face. From the centers we will determine relative distances; from the tracking we will determine a bounding ellipse and angle for the hand. We will also try Viola and Jones's object detection algorithm. We will explore additional useful data from the H.264 encoder, such as the number of reference frames used to find a good match (more might indicate greater motion). We expect these features, together with the more accurate motion vector information due to tracking, to produce a better classifier.

We will then implement AdaBoost and compare its performance to our support vector machine and hidden Markov model. We will try all these algorithms on finger spelling, which will require three classes for the support vector machine and AdaBoost. The AdaBoost papers contain some support for using "confidence" information to train the classifier. We will investigate combining the three techniques, e.g. by feeding the results of the support vector machine into the AdaBoost algorithm.

Our earlier work investigated the intelligibility of raising the frame rate for finger spelling but did not explore the optimal frame rate for finger spelling. While we found no increase in intelligibility, reviews pointed out that the kind of finger spelling we tested was the easiest to recognize from context. We plan to conduct a user study to determine the best frame rate for finger spelling, given bit rate constraints.

4.2 Detecting when frames can be dropped in general

Given positive results on detecting signing, not signing, and finger spelling, we will explore detecting movement epenthesis with the same methods. Though we are not interested in sign language recognition in general, recognizing movement epenthesis is equivalent to sign segmentation, and could be used in a larger system.

More applicable to our purposes is the ability to only transmit signs that matter. We found in previous research that a sign language video at 5 frames per second, regularly spaced, is unintelligible. However, the frame rate may be dropped that low if the frames transmitted are selected intelligently. Parish et al. compared dropping frames at regular intervals to only selecting the beginning and ending frames of the signs [40]. They found that the sequence was more intelligible with only the frames at the beginning and ending of each sign. If we could do this in real time on the cell phone, the savings would be immense. There would be several engineering problems to overcome. First, each frame individually would likely need more bits, because the search space for the motion compensation would be large. Second, we would have to run the compression algorithm on frames that we were not going to send, to obtain feature information. This would still result in savings, since the most expensive processes are sending the data and playing the video, and less frames would be sent for playback. We could also address this problem by choosing different features that did not require compression.

Finally, we would need to be sure that the users did not find the jumpiness of dropping frames during signing so irritating that they would not use the phone. To that end, we plan on conducting a user study with the beginning and ending frames chosen by hand. If the results justify dropping frames for movement epenthesis, we would continue to apply our techniques to the problem of sign segmentation.

4.3 Timeline

October 2007 - March 2008: Continue work on recognizing signing, not signing, and finger spelling, with the aim of submitting to the International Conference on Automatic Face and Gesture Recognition.

1. Implement Kalman filters for face and hand tracking, to obtain better features.
2. Implement Viola and Jones's object detection for tracking.
3. Improve hidden Markov model; experiment with alternate modeling.
4. Evaluate three class support vector machine.
5. Implement a boosted cascade of classifiers via AdaBoost.
6. Experiment with combining these techniques.

April 2008 - May 2008: Run user study to evaluate optimal frame rate for technical fingerspelling. Continue work on machine learning techniques, with the aim of submitting to ASSETS.

June 2008 - December 2008: Apply techniques to the problem of sign segmentation.

1. Evaluate feature set and explore new possibilities.
2. Conduct a user study to evaluate intelligibility of dropping frames during movement epenthesis.
3. Continue to improve machine learning techniques; implement combination via decision trees. Given good results with novel techniques, consider submitting to a pattern recognition, computer vision, or machine learning conference.

Early 2009: Complete dissertation and defend.

5 Conclusion

We have presented an overview of the methodologies from sign language recognition that will be useful in activity analysis of sign language video, and detailed the machine learning techniques that we can employ. We will use these to recognize signing, not signing, and finger spelling, and ultimately to segment continuous sign language video into isolated signs. The overall goal of the project is to increase accessibility for the Deaf, and to that end we will keep the users at the forefront, by performing user studies to ensure that our methods will result in a viable product.

References

[1] S. Akyol and U. Canzler. An information terminal using vision based sign language recognition. In *ITEA Workshop on Virtual Home Environments*, pages 61–68, 2002.

[2] M. Assan and K. Grobel. Video-based sign language recognition using hidden markov models. In *Proceedings of Gesture Workshop*, pages 97–109, 1997.

[3] B. Bauer and K.-F. Kraiss. Video-based sign recognition using self-organizing subunits. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 434–437, 2002.

[4] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363, 1967.

[5] N. D. Binh, S. Enokida, and T. Ejima. A new approach dedicated to real-time hand gesture recognition. In H. R. Arabnia, editor, *Proceedings of the 2006 International Conference on Image Processing, Computer Vision, & Pattern Recognition, Las Vegas, Nevada, USA, June 26-29, 2006, Volume 2*, pages 481–488. CSREA Press, 2006.

[6] N. D. Binh, E. Shuchi, and T. Ejima. Real-time hand tracking and gesture recognition system. In *Proceedings of International Conference on Graphics, Vision and Image Processing(GVIP-05)*, pages 362 – 368, Dec. 2005.

[7] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference on Computer Vision*, volume 1, pages 390–401, 2004.

[8] A. Cavender, R. E. Ladner, and E. A. Riskin. MobileASL: Intelligibility of sign language video as constrained by mobile phone technology. In *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 71–78, New York, NY, USA, 2006. ACM Press.

[9] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image Vision Computing*, 21(8):745–758, Aug. 2003.

[11] N. Cherniavsky, A. C. Cavender, R. E. Ladner, and E. A. Riskin. Variable frame rate for low power mobile sign language communication. In *Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, New York, NY, USA, 2007. ACM Press. To appear.

[12] F. Ciaramello and S. Hemami. 'Can you see me now?' An objective metric for predicting intelligibility of compressed American Sign Language video. In *Human Vision and Electronic Imaging 2007*, January 2007.

[13] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[14] Y. Cui and J. J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, May 2000.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B*, 39(1):1–38, 1977.

[16] R. Erenshteyn, P. Laskov, R. Foulds, L. Messing, and G. Stern. A recognition approach to gesture language understanding. In *International Conference on Pattern Recognition*, pages III: 431–435, 1996.

[17] G. Fang, W. Gao, X. Chen, C. Wang, and J. Ma. Signer-independent continuous sign language recognition based on SRN/HMM. In I. Wachsmuth and T. Sowa, editors, *Gesture Workshop*, volume 2298 of *Lecture Notes in Computer Science*, pages 76–85. Springer, 2001.

[18] G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, Mar. 1973.

[19] R. A. Foulds. Biomechanical and perceptual constraints on the bandwidth requireme nts of sign language. In *IEEE Trans. On Neural Systems and Rehabilitation Engineering*, volume 12, pages Vol I: 65–72, March 2004.

[20] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.

[21] W. Gao, J. Ma, J. Wu, and C. Wang. Sign language recognition based on HMM/ANN/DP. *IJPRAI*, 14(5):587–602, 2000.

[22] GSMA. General packet radio service. `http://www.gsmworld.com/technology/gprs/class.shtml`, 2006.

[23] H. Hienz, K. Grobel, and G. Offner. Real-time hand-arm motion analysis using a single video camera. In *International Conference on Automatic Face and Gesture Recognition*, pages 323–327, 1996.

[24] E.-J. Holden and R. A. Owens. Visual sign language recognition. In R. Klette, T. S. Huang, and G. L. Gimel'farb, editors, *Theoretical Foundations of Com-*

puter Vision, volume 2032 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2000.

[25] C. L. Huang and W. Y. Huang. Sign language recognition using model-based tracking and a 3D hopfield neural-network. *Machine Vision and Applications*, 10(5-6):292–307, Apr. 1998.

[26] C. L. Huang and S. H. Jeng. A model-based hand gesture recognition system. *Machine Vision and Applications*, 12(5):243–258, 2001.

[27] K. Imagawa, H. Matsuo, R. Taniguchi, D. Arita, S. Lu, and S. Igi. Recognition of local features for camera-based sign language recognition system. In *International Conference on Pattern Recognition*, pages Vol IV: 849–853, 2000.

[28] International Telecommunication Union. Trends in Telecommunication Reform 2007: The Road to NGN, Sept 2007.

[29] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *International Conference on Computer Vision*, pages 107–112, 1998.

[30] B. F. Johnson and J. K. Caird. The effect of frame rate and video information redundancy on the perceptual learning of American Sign Language gestures. In *CHI '96: Conference companion on Human factors in computing systems*, pages 121–122, New York, NY, USA, 1996. ACM Press.

[31] M. W. Kadous. Machine recognition of auslan signs using powergloves: Towards large-lexicon recognition of sign language. In *Workshop on the integration of Gesture in Language and Speech*, pages 165–174, 1996.

[32] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[33] T. Kobayashi and S. Haruyama. Partly-hidden markov model and its application to gesture recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 3081–3084, 1997.

[34] S. S. Kuo and O. E. Agazzi. Visual keyword recognition using hidden markov models. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 329–334, 1993.

[35] R. H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *International Conference on Automatic Face and Gesture Recognition*, pages 558–567, 1998.

[36] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proceedings of the British Machine Vision Conference (BMVC) 2002*. British Machine Vision Association, 2002.

[37] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *SIGCHI Conference Proceedings*, pages 237–242, 1991.

[38] E. J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *International Conference on Automatic Face and Gesture Recognition*, pages 889–894, 2004.

[39] S. C. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), June 2005.

[40] D. H. Parish, G. Sperling, and M. S. Landy. Intelligent temporal subsampling of american sign language using event boundaries. *Journal of Experimental Psychology: Human Perception and Performance*, 16(2):282–294, 1990.

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

[42] G. Raetsch, T. Onoda, and K.-R. Mueller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[43] J. L. H. Rebollar, N. Kyriakopoulos, and R. W. Lindeman. A new instrumented approach for translating american sign language into sound and text. In *International Conference on Automatic Face and Gesture Recognition*, pages 547–552, 2004.

[44] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, N.J., 1995.

[45] H. Sagawa and M. Takeuchi. A method for recognizing a sequence of sign language words represented in a japanese sign language sentence. In *International Conference on Automatic Face and Gesture Recognition*, pages 434–439, 2000.

[46] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, Dec. 1999.

[47] G. Sperling, M. Landy, Y. Cohen, and M. Pavel. Intelligible encoding of ASL image sequences at extremely low information rates. In *Papers from the second workshop Vol. 13 on Human and Machine Vision II*, pages 256–312, San Diego, CA, USA, 1986. Academic Press Professional, Inc.

[48] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[49] W. C. Stokoe. *Sign Language Structure: An Outline of the Visual Communication System of the American Deaf*. Studies in Linguistics: Occasional Papers 8. Linstok Press, Silver Spring, MD, 1960. Revised 1978.

[50] S.Young, J.Jansen, J.Odell, D.Ollason, and P.Woodland. *The HTK Book*. Entropic Cambridge Research Laboratory, Cambridge, England, 1995.

[51] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988.

[52] N. Tanibata, N. Shimada, and Y. Shirai. Extraction of hand features for recognition of sign language words. In *Proceedings of International Conference on Vision Interface*, pages 391–398, 2002.

[53] C. Valli and C. Lucas. *Linguistics of American Sign Language: An Introduction.* Gallaudet University Press, 1992.

[54] P. Vamplew. *Recognition of Sign Language Using Neural Networks.* PhD thesis, University of Tasmania, 1996.

[55] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages I:511–518, 2001.

[56] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[57] C. Vogler. *American Sign Language Recognition: Reducing the Complexity of the Task with Phoneme-Based Modeling and Parallel Hidden Markov Models.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Dec. 2002.

[58] M. B. Waldron and S. Kim. Isolated ASL sign recognition system for deaf persons. *IEEE Transactions on rehabilitation engineering*, 3(3):261–271, 1995.

[59] C. Wang, W. Gao, and S. Shan. An approach based on phonemes to large vocabulary chinese sign language recognition. In *International Conference on Automatic Face and Gesture Recognition*, pages 393–398, 2002.

[60] M. H. Yang, N. Ahuja, and M. Tabb. Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, Aug. 2002.

[61] R. Yang and S. Sarkar. Detecting coarticulation in sign language using conditional random fields. In *International Conference on Pattern Recognition*, pages 108–112. IEEE Computer Society, 2006.

[62] Q. Yuan, W. Gao, H. Yao, and C. Wang. Recognition of strong and weak connection models in continuous sign language. In *International Conference on Pattern Recognition*, pages I: 75–78, 2002.

[63] J. Zieren, N. Unger, and S. Akyol. Hands tracking from frontal view for vision-based gesture recognition. In L. J. V. Gool, editor, *Proceedings of Pattern Recognition, 24th DAGM Symposium, Zurich, Switzerland, September 16-18, 2002*, volume 2449 of *Lecture Notes in Computer Science*, pages 531–539. Springer, 2002.