24-787 Artificial Intelligence and Machine Learning for Engineering Design
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
Project Report 2012

AMERICAN SIGN LANGUAGE RECOGNITION SYSTEM

Jason Atwood
Carnegie Mellon University
Pittsburgh, PA, USA
jatwood@cmu.edu

Matthew Eicholtz
Carnegie Mellon University
Pittsburgh, PA, USA
meicholt@andrew.cmu.edu

Justin Farrell
Carnegie Mellon University
Pittsburgh, PA, USA
justin.v.farrell@gmail.com

ABSTRACT

Sign language translation is a promising application for vision-based gesture recognition methods, in which highly-structured combinations of static and dynamic gestures correlate to a given lexicon. Machine learning techniques can be used to create interactive educational tools or to help a hearing-impaired person communicate more effectively with someone who does not know sign language. In this paper, the development of an online sign language recognizer is described. The scope of the project is limited to static letters in the American Sign Language (ASL) alphabet. Two machine learning approaches were implemented: (1) a single hidden layer neural network and (2) a principal component analysis (PCA) model. In the former case, images were processed to reduce the number of pixels (input nodes to the network) while maintaining an appropriate amount of variance between signs. Over-fitting was avoided using *k*-fold cross validation (*k=2*). The PCA model facilitated reduced dimensionality without loss of relevant information (e.g. from scaling or normalization). The results indicate that both approaches recognize signs effectively for subjects included in the training process (>95%), while untrained subjects produce poor accuracy (~40-70%). When all subjects were included in the training set, the best neural network exhibited 95.8% accuracy compared to 96.1% accuracy for the PCA model. Custom MATLAB user interfaces were created for acquiring training samples and for testing the machine learning approaches on live data streamed from a webcam. Despite high error for unseen subjects in offline processing, the system is able to recognize all letters in the real-time GUI simply by adjusting the hand position or orientation. Future improvements include incorporating a dynamic bounding box, lifting the restrictions on scaling/ rotation/background noise, and recognition of dynamic letters and two-handed words.

INTRODUCTION

Real-time gesture recognition has been highly researched over the past two decades, with many human-computer interface applications ranging from virtual reality to sign language translation [1]. Within the latter domain, researchers have implemented both vision-based [2-9] and sensor-based (e.g. instrumented gloves) [10-13] recognition. While both methods have produced favorable results, vision-based recognizers do not require hardware to be worn by the user, producing a more natural feel. Among both architectures, neural networks are a popular machine learning technique for sign language recognition [4-6,9-11,12].

American Sign Language (ASL) consists of both static and dynamic gestures. For continuous communication, hand shape, movement, and location (often referred to as a "chereme" [8]) are crucial features during translation. The complexity introduced by temporal information lends itself well to techniques such as hidden Markov models [7], recurrent neural networks [6] (also time-delayed [9]), or a combination of both [10]. More generally, dynamic gestures favor feature extraction rather than image-based input to recognition models.

A much simpler task is to limit the scope of the system to static gestures. Perhaps the most obvious application is finger-spelling, which involves the discrete set of 26 signs corresponding to the letters of the alphabet (note that two letters, J and Z, are dynamic). The authors have developed a computational tool for online recognition of the static letters in the alphabet using image-based neural networks and principal component analysis (PCA). The following sections detail the methods used for acquiring data samples, processing images, implementing the aforementioned machine learning techniques, and creating the user interface. Results are presented and discussed along with potential improvements for the future.

TECHNICAL APPROACH

Data from three subjects was collected and utilized to construct a neural network and principle component model. Each subject generated 20 samples of each static letter (480 total samples). The data was subsequently used to evaluate the performance of both models.

Image Acquisition

A custom user interface (see **Figure 1**) was created using MATLAB software to facilitate efficient acquisition of training samples.

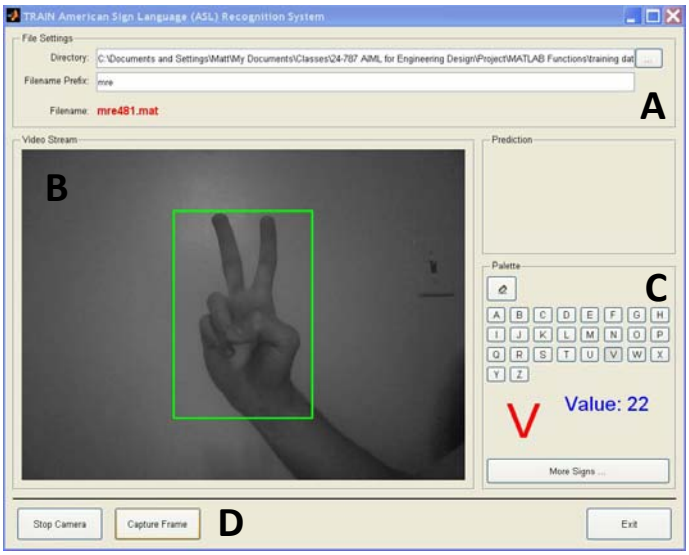

Figure 1. Custom GUI for acquiring sample images. **(A)** File settings panel includes self-selected directory and filename prefix (e.g. initials) options. File number increments automatically. **(B)** Live video stream from webcam. Green bounding box is fixed. **(C)** User selects appropriate label from the sign palette. Each letter corresponds to a number (1-26). **(D)** Buttons for controlling camera, image capture, and exiting the GUI.

Image Processing

One of the major challenges for image-based recognition is optimization of image processing protocol. Over-processing (specifically with regard to normalization and scaling), while reducing image dimensionality, can lead to poor variance between signs. Under-processing, by contrast, allows for easy differentiation of signs but increases dimensionality. Finding the right balance is crucial for obtaining high accuracy while reducing computational complexity. For this project, combinations of grayscale conversion, cropping, scaling, and binary conversion were employed for each of the machine learning methods. **Figure 2** illustrates the processing results on a sample image.

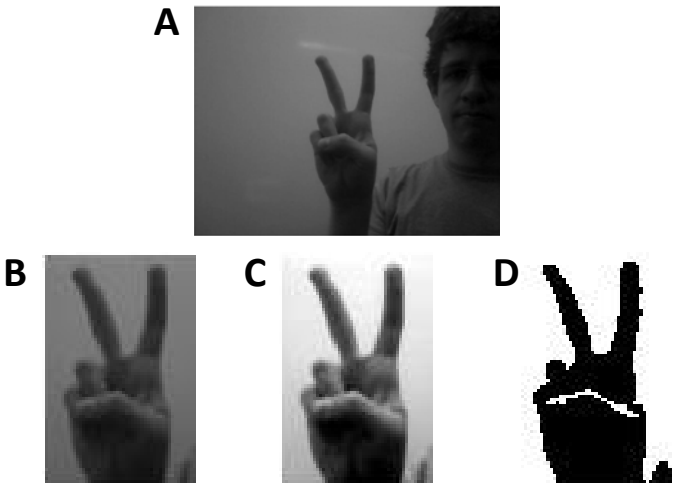

Figure 2. Image processing routine on sample image (letter V). **(A)** Original image. **(B)** Cropped and scaled. **(C)** Sharpened for contrast. **(D)** Converted to binary.

Neural Network

A neural network was constructed to recognize ASL letters. The appropriate structure parameters and perceptron weights were selected based on a comparison method. Initially, several networks were generated with varied number of hidden nodes and training data size. Hidden nodes numbered 10, 50, 100, 250, or 500. The training data consisted of 120, 480, or 960 sample points. Each network was limited to a single hidden layer, a single number of input and output nodes, a max epoch of 100, and feed-forward node interconnections. To avoid over-fitting the data, *k*-fold cross validation was implemented with $k=2$, such that half of the training set was not used to train the network.

The neural networks were trained using a selection of the 480 samples, after the images had been cropped, scaled, sharpened, and binary filtered to a 50x75 pixel image. The image pixels were then arranged into a single column vector, corresponding to 3,750 input nodes to the network. The alphabetic output of the network was represented by 26 binary nodes, each corresponding to one of 26 letters. The structure of these neural networks is shown in **Figure 3**. The resulting 15 networks were ranked based on their performance on validation data. The best-performing neural network was selected for implementation of real-time ASL translation.

The above process was implemented four times with varied training data sets. In three instances, one test subject's data was removed from the training set, resulting in a network trained on two subjects. To generate the fourth and final network, all subjects contributed equal amounts of training data.

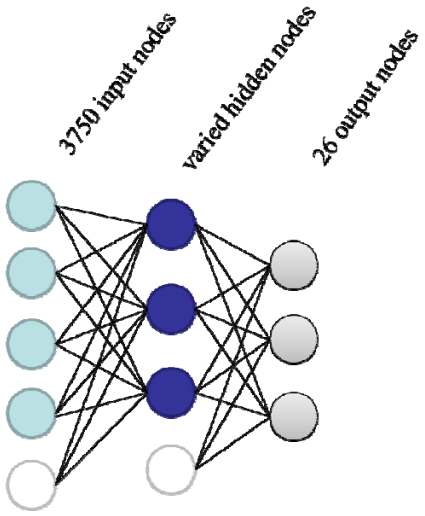

Figure 3. Neural network structure consisting of a single input layer, hidden layer, and output later. The number of hidden nodes and training samples were varied to find the best-performing network.

Principle Component Analysis

For comparison to the neural network, PCA with a nearest-neighbor classifier was used for sign recognition. While scaling images to reduce the pixel count is one method of reducing the

dimensionality of an image, it results in a substantial amount of information loss, with no regard for the importance of that information. By comparison, PCA reduces the dimensionality of the problem through a selective elimination the least important features.

Each 200x300 cropped grayscale training image was converted to a 60,000x1 image vector, Γ_i. For training samples used (960 total), the mean-subtracted vectors were compiled into matrix $A^{60,000 \times 960}$, where

$$\mathbf{A} = [\Gamma_1 - \mu, \Gamma_2 - \mu, ..., \Gamma_{960} - \mu,] \quad \text{with} \quad \mu = \frac{1}{960}\sum_{i=1}^{960}\Gamma_i$$

The set of eigenvectors, v_i, of A^TA determine the linear combination of the training images that form the *eigenhands*, U_i.

$$U_i = \sum_{k=1}^{960} v_{ik}(\Gamma_i - \mu) \quad \text{for eigenhand i=1, 2,…, 960}$$

Together, the eigenhands form a basis for the *hand space*, where any sign can be represented as the linear combination of eigenhands. Each original training image was projected into the hand space where, without any loss of data, the dimensionality of the vector was reduced from 60,000 to 960. Discarding features corresponding to lower eigenvalues can further reduce the dimensionality. **Figure 4** shows the first three eigenhands of the hand space. The coefficients, a_i, are the weights corresponding to each eigenhand that represent an image in the hand space.

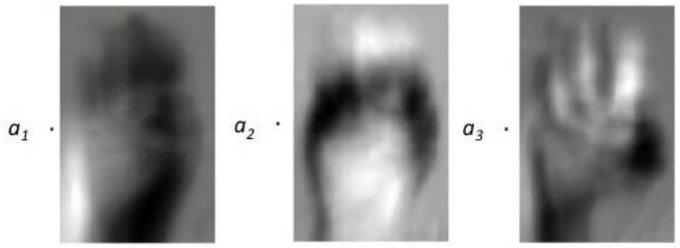

Figure 4. Illustration of the first three eigenhands in the hand space.

When a new test image is introduced, it is projected into the hand space where it is represented as a linear combination of the eigenhands. Once in the hand space, the nearest-neighbor classifier is used to label the new sign as a specific letter. This is accomplished by finding the training image with the minimum Euclidean distance to the test image. Although the nearest-neighbor search can be done in the original $R^{60,000}$ space, the computational cost is too high to perform online. By applying PCA and projecting images into the hand space, the computation is fast enough to perform at least one recognition per second.

Online Recognition

The results from the neural network and PCA approaches described above are utilized for real-time sign language recognition using a custom GUI (not pictured). The user interface is similar to the training acquisition GUI shown in **Figure 1**, with the exception that the top panel is used for selection of a neural network or PCA, the sign palette is supplanted with the output weights for each letter (neural network case only), and the prediction panel shows the recognized letter. The prediction rate is set to update once per second. The same static bounding box is used as before.

RESULTS

Neural Network

The structural parameters of the best performing network in each of the four training cases are listed in **Table 1** along with performance measures. *Training accuracy* refers to the percentage of trained images that were recognized correctly. *Validation accuracy* reports the percentage of validation (untrained) images that were recognized correctly. The three networks that were trained on only two subjects were tested against the third subject (see **Table 2**).

Table 1. Best performing neural networks.

Training data	Subjects 1,2	Subjects 1,3	Subjects 2,3	All subjects
Hidden nodes	250	100	100	100
Training size	960	960	960	1440
Training accuracy (%)	100	100	100	100
Validation accuracy (%)	97.3	98.8	95.4	95.8

Table 2. Neural network response to new subjects.

Training data	Subjects 1,2	Subjects 1,3	Subjects 2,3
Test data	Subject 3	Subject 2	Subject 1
Test accuracy (%)	57.5	43.8	48.8

Principle Component Analysis

The performance of the PCA model was tested in two ways, both of which use only the 24 static signs in the ASL alphabet. First, the model was trained by all three subjects, and was tested on unseen samples from those same subjects. The results are given in **Table 3**.

Table 3. PCA accuracy on unseen data.

Training data	All subjects
Test data	All subjects
Test accuracy (%)	96.1

For the second test, the model was trained on two subjects and subsequently tested on the third, unseen subject. The results are given in **Table 4**.

Table 4. PCA accuracy on unseen subjects.

Training data	Subjects 1,2	Subjects 1,3	Subjects 2,3
Test data	Subject 3	Subject 2	Subject 1
Test accuracy (%)	71.5	45.2	62.7

DISCUSSION

The results demonstrate that a low-dimensionality neural network can accurately recognize static letters in the ASL alphabet when the user is included in the training of the neural network. However, even the best performing network had difficulty recognizing signs when introduced to a new test subject. A probable source of error is the static bounding box used in image acquisition. Accuracy decreases if the test subject has a consistently different hand position or orientation within the bounding box compared to the training subjects. Hand size also varies among subjects, which needs to be normalized in the image processing routine to improve performance. Despite large errors in offline processing of test subjects, the online GUI was able to correctly predict every static letter for test subjects with minor alterations in scale, position, and orientation of the hand within the bounding box.

For the first PCA test, in which the model was tested on unseen data from seen subjects, the accuracy was very high. In this test, the variability between users' signing technique as well as hand placement is eliminated. The largest variability is the lack of repeatability of a single subject to display a given sign. When the model is tested on an unseen subject, the accuracy is reduced drastically. This, to some extent, is expected from a PCA technique unless further steps are taken to normalize the orientation and scale of the images. Since PCA is sensitive to rotation, translation, and scaling, the introduction of a new subject with an individual technique for the sign causes issues.

Despite successful implementation of both a neural network and PCA model, the system could benefit from several improvements. One method to rectify translation, scale, and rotation issues is to add a dynamic bounding box for image capture. This technique would locate the hand during video streaming, perhaps in a similar manner to previous methods for tracking dynamic hand motions [14], and frame the sign such that orientation and scale are uniform across all images. Another solution is to create large amounts of training data from few samples by artificially altering translation, rotation, and scale effects, much like an approach used in symbol recognition [15]. Other improvements include the recognition of dynamic letters and words, which add a temporal component to the problem. In the short term (i.e. to detect letters J and Z), output nodes can be added to the neural network for static images produced sequentially when signing those letters. The online system would then wait for the proper sequence of outputs (e.g. 10-27-28 could represent J and 26-29-30 could represent Z) before displaying the predicted sign. More intelligent options that have already been explored include recurrent neural networks [6] and hidden Markov models [7].

CONCLUSIONS

Neural networks and principal component analysis are two powerful machine learning approaches for real-time sign language recognition. Image-based input is successful for static signs, although a feature-based approach may be better for dynamic gestures. The recognition techniques presented here can have a broad impact on future human-computer interaction applications.

REFERENCES

[1] Wu Y and Huang TS. Vision-based gesture recognition: a review. *Lect Notes Comput Sci* **1739**, 1999.

[2] Charayaphan C and Marble AE. Image processing system for interpreting motion in ASL. *J Biomed Eng* **14**, 1992.

[3] Derpanis KG, Wildes RP, and Tsotsos JK. Definition and recovery of kinematic features for recognition of American sign language movements. *Image Vision Comput* **26**, 2008.

[4] Huang CL and Huang WY. Sign language recognition using model-based tracking and a 3D Hopfield neural network. *Mach Vision Appl* **10**, 1998.

[5] Munib Q, Habeeb M, Takruri B, and Al-Malik HA. ASL recognition based on Hough transform and neural networks. *Expert Syst App* **32**, 2007.

[6] Murakami K and Taguchi H. Gesture recognition using recurrent neural networks. *Proc SIGCHI Conf on Human Factors in Computing Syst*, New Orleans, LA, pp. 237-242, 1991.

[7] Starner T, Weaver J, and Pentland A. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Trans Pattern Anal Mach Intell* **20**, 1998.

[8] Tamura S and Kawasaki S. Recognition of sign language motion images. *Pattern Recogn* **21**, 1988.

[9] Yang MH, Ahuja N, and Tabb M. Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Trans Pattern Anal Mach Intell* **24**, 2002.

[10] Fang G, Gao W, Chen X, Wang C, and Ma J. Signer-independent continuous sign language recognition based on SRN/HMM. *Lect Notes Comput Sci* **2298**, 2002.

[11] Fels SS and Hinton GE. Glove-TalkII – a neural network interface which maps gestures to parallel formant speech synthesizer controls. *IEEE Trans Neural Netw* **9**, 1998.

[12] Hernandez-Rebollar JL, Lindeman RW, and Kyriakopoulos N. A multi-class pattern recognition system for practical finger spelling translation. *4th IEEE Intl Conf Multimodal Interfaces*, Pittsburgh, USA, pp. 185-190, 2002.

[13] Waldron MB and Kim S. Isolated ASL sign recognition system for deaf persons. *IEEE Trans Rehabil Eng* **3**, 1995.

[14] Rehg JM and Kanade T. Model-based tracking of self-occluding articulated objects. *Proc 5th Intl Conf Comput Vision*, Cambridge, MA, pp. 612-617, 1995.

[15] Fu L and Kara LB. Neural network-based symbol recognition using a few labeled samples. *Computers & Graphics* **35**, 2011.