Sign Language to Text Conversion (July 2012)

James M.D. Power (200715167), Grant E. O'Brien (200831543), *MUN*

Abstract— Communication between those who cannot speak vocally and those who can is a challenge faced by many people. Our Machine Vision project looks to overcome this barrier by converting sign language hand gestures to English text. We will outline our hardware setup, showing that this barrier can be overcome with just a camera, a glove, and a laptop. We will also explain the algorithms and graphical interfaces that are used in our system. Finally, we will explain the challenges and limitations that we faced.

***Index Terms* — Image Processing, Image analysis, Object Recognition, Machine Vision**

I. Introduction

For people who cannot communicate vocally, sign language is a powerful tool they can use to overcome the communication barrier. However, if the listener does not understand sign language, the barrier has not been broken. Thus, there is a need for a system to interpret sign language and convert it to text. This report will outline the hardware implementation of our solution, with accompanying diagrams. Furthermore, this document will also discuss some of the software implementation, with accompanying diagrams for both sections.

II. Research/Scope

A. Sign Language

Sign language is a form of non-verbal communication using hand gestures. There are multiple sets of sign language gestures, including different gestures for different spoken languages [1]. This project will focus on only variant of sign language, the American Sign Language (ASL) standard.

ASL does not directly convert English words into hand gestures. Instead, it converts *expressions* into gestures. An example would be the English words *right* and *write*. In English, they have the same pronunciation, but in ASL they each have different hand gestures. Despite having gestures for expressions, ASL also has separate expressions for each individual letter of the English alphabet. Furthermore, ASL also utilizes facial expressions to convey certain types of meaning.

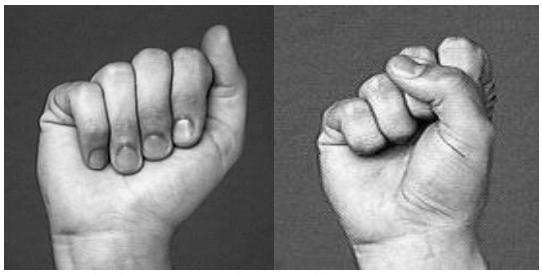


Fig. 1. The letters 'A' and 'S', as displayed using ASL.

B. Scope

Our project will not try to convert all the gestures to their proper textual representation. Instead, our project focused on trying to convert the twenty-six (26) gestures for each letter of the alphabet to their textual representation. Once this system is in place, a user could directly spell out whatever word they wish using individual letters.

III. Comparable Systems

Our research has led to the discovery of many existing systems that either convert sign language to text, or perform the opposite. The comparable systems are discussed below, with a reference to find further information.

A. Hand in Hand: Automatic Sign Language to English Translation- RWTH Aachen University, Dublin City University

This project is very similar to our project [2]. It aims to convert ASL and Irish Sign Language to English speech, also using computer vision. To make this interpretation, the researchers behind the project analyze hand gestures, body movements, and facial expression. In a sense, our project has a smaller scope, as our project will focus only on the ASL standard, and it's 26 letters.

B. Computer Vision for Recognition of American Sign Language- Purdue University

This is also very similar to our project, however this project is also similar to the one listed above [3]. However, their project will utilize a set of *recognition procedures*, whereas our project will utilize a set of training images. This means that our project will focus more on the shape and orientation of the hand, and less on movement.

C. *Say It Sign It (SISI)- IBM*

IBM's project is actually the exact opposite of our project [4]. This system converts speech from the user to sign language, using the British Sign Language system. The gestures are *signed* back to the user via a digital avatar.

D. *An Image Processing Technique for Translation of ASL Finger-Spelling to Digital Audio or Text- Rochester Institute of Technology*

While very similar to our project, it is also slightly different [5]. It will translate ASL finger spelling into English text, however, they intend to use a more pure image processing approach. They have decided against using any sort of tracking utilizing a glove.

E. *Accelespell Kit- Institute for Disabilities Research and Training, Inc.*

Unlike the solution proposed by the Rochester Institute of Technology's solution, the Accelespell Kit *will* utilize a glove [6]. The glove will be wired and calibrated, allowing for the modeling of hand gestures by the individual wearing the glove. By using a glove, the Accelespell Kit is trying to avoid using a computer vision system.

IV. Our Approach

The hardware setup for this project involved multiple components working in tandem. Our goal was to constrain the system so that the detection algorithms can be made simple.

First, there is a digital camera that is connected to a laptop computer running MATLAB. A live, grayscale video stream is being sent to this computer, and is displayed on screen as part of a MATLAB program (this program will be discussed in later sections). Once every few seconds, this program will grab a screen shot from the camera's live stream.

The user wears a black glove, with strategically placed stickers on it (also discussed later). The user will place the glove wearing hand in front of a white screen, and will perform the sign language gestures in front of it. It is preferable that the user does not enter the camera view, meaning they should extend their arm out if possible. The user may utilize a second computer, so that the sign language gestures could easily be viewed and signed.

A screen shot of the full setup while being used is available in figure 2. This setup was used for both capturing training images, and running the completed sign language detection system.

Fig. 2. Our complete hardware setup, with James signing a 'B'.

V. The Glove

The glove utilized in our hardware setup has been carefully designed to maximize it's effectiveness. The purpose of the glove was twofold: first, it was meant to give our camera something to track; second, it was meant to provide an easy way to distinguish between sign language gestures that were similar to each other. As seen in figure 1, the gestures for 'A' and 'S' are quite similar to one another, and simply trying to detect the gesture may not be enough.

The first purpose was achieved through the use of proper materials and colors, as well as our algorithms (discussed in the next section). A black glove was used in conjunction with a white background so that the background could easily be removed, leaving only the glove.

The second purpose was achieved through the use of specially placed white stickers on the glove. These stickers were placed on the fingernails of the glove. This placement was effective because many of the similar gestures differ in how the fingernails/fingers are shown. As seen in figure 1, the letter 'A' exposes the fingernails, while 'S' requires the fingers to be curled, thus hiding the nails. The stickers were placed on alternating fingers; if we had stickers on each finger, the stickers could overlap on gestures such as 'S'. This could cause problems when thresholding, because the stickers could be seen as one large hole. A sticker was placed on the palm to assist in differentiating between letters that cover the palm.

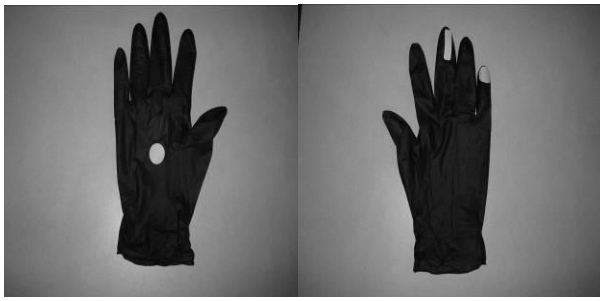


Fig. 3. The two sides of the glove used to help detect gestures.

VI. Algorithms

By constraining our system to the setup previously described, we have transformed the complex sign language detection problem into a straight-forward blob analysis problem. The MATLAB program that we used involves three major types of algorithms in the detection process: the image clean-up algorithms, the object detection / form parameter calculation algorithm and the object classifier algorithm.

A. Image Clean-Up

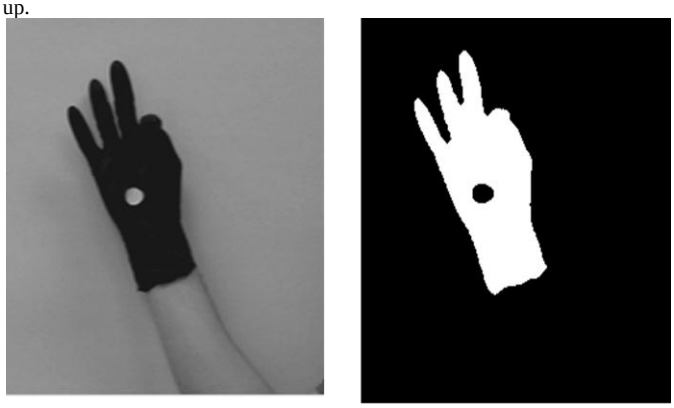
Our image clean-up algorithm is catered specifically for our project setup. It involves thresholding, noise reduction and shaping with morphological operators. A sample result of this algorithm is shown in the figure below.

Thresholding is performed using a manual threshold, which is set at 30. This low threshold was found experimentally, and is effective in separating the hand from the arm and the rest of the background. Initially, we attempted to calculate our threshold based on otsu's method, but we found that this didn't fully remove the arm from the image, as the histogram was not bi-modal.

Noise reduction is performed using morphological opening and closing with a 2-pixel diameter disk structuring element. This effectively removes small spots on the image and also smooths the border pixels around the hand.

Further shaping is performed with the 'bwareaopen' and 'bwareaclose' MATLAB functions. These functions perform morphological opening and closing on areas in the image that are smaller than a given number of pixels. We use these functions with an area of twenty-five pixels in order to remove unwanted holes from the hand (i.e. small spaces between the fingers).

Fig. 4. The frame retrieved from the camera, before and after being cleaned up.

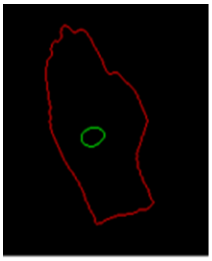


Before Clean-Up After Clean-Up

B. Object Detection / Form Parameter Calculation

We perform object detection using by using the technique of contour tracking. Basically, we iterate through each pixel in the image, and if we encounter a black-to-white transition, we know that we have encountered an object. We then call a function that performs the contour tracking using crack code.

While calculating the crack code, our function calculates perimeter and moments up to M_{11} using a switch statement on the current crack code number. Once these parameters are calculated, we calculate several form parameters that are independent of position, orientation, reflection and sometimes scale. As well, when we find an object with a negative area, we detect it as a hole and associate it with an object. A sample table of calculated form parameters is shown in the next figure. Our reasoning for including form parameters that were not independent of scale was the fact that our set-up was always constrained to a distance of one meter between the camera and the backboard. In testing we confirmed that these form parameters were indeed accurate in describing our symbols.



Holes	Comp	Phi1	Phi2
1	31.86	1.46e7	8.27e13

Imin	Imax	Ri	Psi1	Psi2
2.75e6	1.18e7	0.2324	0.2090	0.0169

Fig. 5. The fully processed hand gesture from figure 4, with form parameters included.

C. Object Classifier

Our object classifier works as a simple 'nearest' neighbour algorithm. It must be passed two sets of data: the form parameters of the object to be detected, and the form parameters from our training set. Both of these sets of data must be normalized before being passed to the detector.

Normalization is done by finding the minimum and maximum value for each form parameter in the training set. Then, the minimum is subtracted from each value, and the result is divided by the maximum. The result is a set of parameters that are scaled from zero to one.

Once the normalized parameters are found, the classifier algorithm performs a Euclidian distance calculation between the parameters from each of the training images and the detected object. This means that the difference in each parameter is calculated, and the total difference is found by performing the square root of the sum of the squares of these differences. Initially, we considered weighting these parameters for more accurate detection, but through experimentation, we determined that none of these parameters provided a significant advantage over the others. Therefore, we did not perform weighting.

VII. Interfaces

We developed three graphical user interfaces (GUIs) that can be used with our sign language detection setup and algorithms. They are the Learning GUI, the Sign Language Recognition GUI and the Image GUI. The first two of these three GUIs are used for gathering training images and benchmark form parameters, and testing the full system, respectively. The third GUI mentioned is used to test our detection algorithm using static images.

A. *The Learning GUI*

Our Learning GUI is the interface used to gather training images. It is controlled by several buttons within the window. When the Learning GUI is initially opened, the axes on the right side of the window will show a preview of the live video stream. A screen capture of the Learning GUI is shown in the next figure.

Fig. 6. The learning GUI.

The user can capture images from the live video stream shown in the window. After image capturing, they can choose to analyze the image, which will perform clean-up, contour tracking and form parameter calculation based on the algorithms previously described. The user may then save the form parameters in a matrix of training image parameters. These parameters are used as benchmark data for later detection of signs.

B. *The Sign Language Recognition GUI*

This is the main GUI that complements our sign language recognition project. The major components of this GUI are a video preview, a count-down timer, and a text output. A screen capture of this GUI is shown in the next figure.

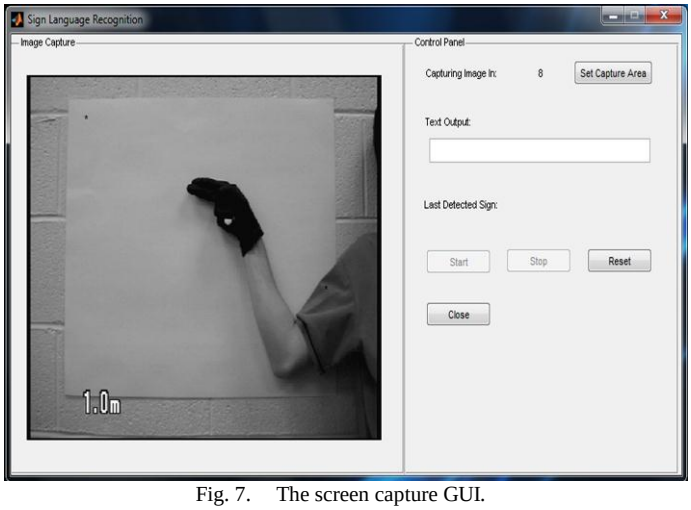


Fig. 7. The screen capture GUI.

The video preview shows the live video feed from the digital camera. When the count-down timer reaches 0, the next frame is captured from the video stream and is processed with clean-up, contour tracking and form parameter calculation. Finally, the calculated form parameters and the training image parameters are passed into the classification algorithm so that the sign can be detected. The name of the detected sign is printed to the text output box, and the timer is reset.

C. *The Image GUI*

The image GUI is a simple GUI which we developed specifically for testing and demoing our classifier algorithm. A screen capture of this GUI is shown below. Within the Image GUI, the user can load an image of a sign language gesture, which will be shown in the image axes. The user can then choose to detect the sign in the image, which will perform clean-up, contour tracking and calculation of form parameters. The form parameters will be passed to the classifier algorithm to detect the sign.

Fig. 8. The Image GUI, used mainly for practice and testing.

VIII. CHALLENGES

The project has encountered several challenges over the course of it's development.

One of the most important challenges faced was trying to isolate the hand from the background and the rest of the body. This was overcome by the use of the glove, and our algorithms.

Another challenge faced was the enormity of the ASL standard. There are gestures for numerous expressions, words, and letters. We narrowed our scope to focus only on the gestures for letters.

The similarity of many symbols also provided a challenge. As noted, the use of the glove allowed us to distinguish between these letters.

The glove in and of itself was also a challenge. The stickers, as noted previously, had to be strategically placed to maximize their effectiveness.

Implementing the form parameters was another challenge. In order to get accurate results from the calculations, the gesturer must perform the gestures in a similar fashion every time. Any deviation could make detection difficult.

Another challenge was the dependency on lighting conditions. If the lighting conditions in the room used to display the technology are different than the conditions in the room used to train the images, there could be detection issues.

IX. Conclusion

While our implementation may not be the most practical, this project was able to create a proof of concept indicating that readily available components could provide a solution to our problem statement. We were able to successfully implement a vision system to convert from sign language to text, with a fairly successful match rate. We did not need to use any expensive equipment to achieve this.

In the future, improvements could be made to our implementation to remove the dependency on lighting conditions. Furthermore, the mobility of our system could be improved, so that in the future, users could potentially carry such a system with them in their everyday lives.

References

[1] D. M. Perlmutter. (2012). *What is Sign Language?* [Online]. Available: http://lsadc.org/info/pdf_files/Sign_Language.pdf

[2] D. Stein et al. "Hand in Hand: Automatic Sign Language to English Translation," RWTH Aachen Universtity, Germany, Dublin City University, Ireland. 2007.

[3] A. Kak et al. (2008). *Computer Vision for Recognition of American Sign Language* [Online] Available: https://engineering.purdue.edu/RVL/Research/ASL/index.html

[4] S. Tomasco. (2007). *IBM Research Demonstrates Innovative 'Speech to Sign Language' Translation System*[Online] Available: http://www-03.ibm.com/press/us/en/pressrelease/22316.wss

[5] C. M. Glenn et al. "An Image Processing Technique for the Translation of ASL Finger-Spelling to Digital Audio or Text," Rochester Institute of Technology, Rochester, NY, 2005.

[6] J. Hernandez-Rebollar et al. "AcceleSpell, a Gestural Interactive Game to Learn and Practice Finger Spelling.," in *ICMI'08*, Chania, Greece. 2008, pp. 189-190

James M.D. Power James Power is a Computer Engineering student at Memorial University, currently in the final year of his studies. Having completed five work terms, James has worked in the public sector for the Department of Transportation and Works, and for Memorial University's Computing and Communications division. He has also completed two work terms in the private sector, for General Dynamics Canada in Calgary, working on communication systems for the Canadian Forces. Inspired by science fiction and video games to pursue a career in technology, James has a focus on web and software development.

Grant E. O'Brien Grant O'Brien is a senior Computer Engineering undergraduate student in the Faculty of Engineering and Applied Science at Memorial University of Newfoundland. Having completed multiple work placements with organisations in the Computer Engineering field, Grant has developed an array of skills including programming and computer application building, communication system design and simulation, circuit design and analysis, and formal report writing. Grant is also involved in various student groups at MUN, including the IEEE Student Branch. Grant holds interest in the fields of image processing, application development and communication systems engineering.