Computer vision and gesture recognition are two rapidly advancing sects of computer science. They aim to allow computer systems to identify visual queues and objects. ASL-to-Text is a video interface program that uses image processing and machine learning techniques to transcribe the beginning of the American Sign Language (ASL) alphabet into English text, taking in visual queues and identifying their English meaning. The goal of our program is to ease communication between Deaf communities and hearing communities by allowing the ASL user to sign and the English speaker to see the textual meanings of those signs. In the future, we plan to increase the amount of letters that the program can process and implement a video-chat interface so that ASL-to-Text can be used for its intended purpose.

**Abstract**

ASL-to-Text is a real-time American Sign Language (ASL) to English text transcriber. The user signs a letter in a webcam’s frame and the program will display the corresponding English letter in a text box.

**Background**

American Sign Language (ASL) was developed as non-verbal communication for the Deaf/Hard of Hearing community. On the other hand, ASL also creates a language barrier between signers and non-signers. To ASL users in the Deaf/Hard of Hearing community, writing English is often their second language. Those users may feel the pressure to adapt to larger society by using English. Therefore, this community feels more comfortable expressing their ideas in their first language: ASL. Currently, the technology available to bridge this gap between signers and non-signers involves a human third party translator, such as Video Relay Service (VRS).

ASL-to-Text a program that aims to bridge the communication gap between the Deaf/Hard of Hearing and Hearing Communities without the assistance of a human interpreter while letting each user speak in their native language.

**Set-Up**

User sits in front of a green screen with black gloves on facing a web camera. User signs a letter and presses the space bar. In the text box, the program will show the translation of the sign.

**Interface**

The screen that they will see has two components:

1. **Video feed of user with a red box indicating where the user should sign**
   - The screen displays the users full upper body because facial gestures are an important feature of ASL.
2. **Text box is located on the right of the video feed**
   - We chose this location because it is regarded as disrespectful to lower your eyes from the face of the person signing. This is because they feel that the audience is ignoring them.

**Image Processing**

1. **Assumptions:** According to our set-up (see Set-Up), we eliminate possible environmental differences by using the green screen and black glove.
2. **Frame Capture:** We used javaCv, a Java wrapper for OpenCV the largest open source computer vision library, to capture single frames from a webcam.
3. **Frame Manipulation:** Using Java, we then translate each frame into grayscale and then downsize it allowing us to make simpler, smaller vectors.
4. **Vectors:** We get the average value (from 1-10) of each pixel in the frame. Those values are then appended in order into a vector. The vectors serve as input for the machine learning algorithms.

**Machine Learning**

The vector of pixels is input into a machine learning program that is written using Waikato Environment for Knowledge Analysis (WEKA). This program uses Sequential Minimal Optimization (SMO), a very powerful learning algorithm which is used during the training of Support Vector Machines (SVMs), to build a model that will predict what class the inputted vector belongs to. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate classes (the training data) are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a class based on which side of the gap they fall on. Once a class is determined for the vector, it is outputted and printed in the interface.

**References**

- ascent.waikato.ac.nz/ml/weka.html
- hastie.visResearch.org/ml-lincoln.pdf
- mlpack.org
- weka.sourceforge.net/doc/weka/summary.html
- http://www.cs.iit.edu/~agam/cs512/lect