ECEN 5593
FINAL PROJECT REPORT

SIGN LANGUAGE INTERPRETER

Project By:
Sairam U J Muttavarapu
Shivasankar Gunasekaran

Executive Summary:

The idea of the project is to help people who don't understand American Sign Language (ASL) to be able to understand it. It also helps people who use sign language to express their ideas easily to a lot of people. This project would use a camera which would capture the images and then processes these images on a PC/microcontroller to figure out what sign it is and finally prints the interpreted output onto the screen Also the interpreted message is converted to speech.

Background:

This idea came to us when we saw a person using sign language in the bus. We saw that he needed an interpreter for him to convey his ideas and expressions. No one can have an interpreter with them all the time. So it struck us that it would be helpful if we create an interpreter which would help all the people involved in the conversation (in which one or more people use sign language) understand each other without the need for a human interpreter.

Initial Idea:

We are planning to use a USB camera and an ARM Cortex A7 or A8 micro-controller which can run an OS like Ubuntu on it. We have two boards in hand which satisfies the requirements and these are:

1. Altera De1-SoC Cyclone V
2. Beagle-bone Black

We are using OpenCV version 2 for image processing. The user should wear a glove, so that the color of the hand of the user isn't a problem. We are planning to implement all the alphabets and the some 3-5 most frequently used phrases. So the Micro-controller will maintain a database of all the signs and the respective interpreted text which acts as a Lookup table. When the user shows a sign in front of the camera, the camera would capture it and process that image using various processing algorithms. Once all the required algorithms are applied, the processed image is compared with the database that we have created. If there is a match, the output would be shown on the website, else an error message would be shown.
In short the hardware and software that we are planning to use are:

Hardware:

1. Micro-controller
    a. Altera De1-SoC / Beaglebone Black
2. USB Camera

Software:

1. OpenCV
2. PHP
3. Python

Actual Implementation:

Hardware:

1. A laptop with Intel i7 processor, 2.4GHz.
2. Altera DE1-SoC
   All the software setup was done, but due to time constraint we didn't port the code onto this platform.

Software:

The code was developed in **Python 2.7** using **OpenCV 3.1.0.** The text to speech conversion was accomplished by using **Pyttsx** library.

Software Explanation:

The basic idea was to develop the code for recognizing the gestures shown by hand and then determine what sign in ASL is being shown. A glove was worn by the user so that there wouldn't be a difference in the image of the hand for people with different complexion.

Initially the hand was detected by extracting the color of the glove from the background. This color extraction was made easier by first converting the RGB image to a HSV image and then using the range of Orange color in the HSV spectrum, the hand was extracted from the background. The grayscale image was obtained by splitting the HSV image into 3 parts. This grayscale image was then

thresholded by using a threshold value and using the Otsu algorithm. Then thresholded image is obtained. This image is then run through morphological transformations like eroding and dilating which were run in 2 iterations. Erode, erodes all the white spots inside a black region, while dilate removes all the black spots in a white region. Applying these transformations on to the thresholded image would give a sharper image of hand eliminating all the noise. Once all the noise was eliminated, this thresholded image was then used to find all the contours in the image. From all the contours detected, the contour with the maximum area was only extracted, assuming that it would be our hand. After determining the contour with the maximum area, this contour is used to draw the convex hull over it. The convex hull is the polygon that wraps the whole contour like a rubber band. Next the convexity defects was also obtained. After detecting the convexity defects, all the minor convexity defects are removed by checking the depth of each convexity defect. If the depth of the convexity defect is between the values of 3000 to 50000, then those points would be considered as the convexity defect in the palm.

Palm Centre Detection:

Next, the centre of the palm was determined. This can be done by identifying any three convexity defect points at a time and then determining whether the minimum enclosing circle of these three points encloses all the other defect points available. If it does then this circle is the minimum enclosing circle. Otherwise, the next set of three points is taken and the above process is repeated. The minimum enclosing circle is found by assuming the two lines connecting the three points to be the chords of a circle. The centre of the circle would be the intersection point of the perpendicular bisectors of the chords formed by the three points. The angle formed by these three points is also noted. Then the radius of the circle formed by three points at a time is determined and sorted in descending order. The angle formed by the points that form the circle with the maximum radius is checked to be less than 90 degrees. If yes then, the minimum enclosing circle is the circle formed by these three points. Else the next highest radius is determined and the above mentioned check is applied on it. If two sets of three points have the same radius, then the set of points with the maximum radius is checked first, followed by the next set of

points. By iterating through all the sets of defect points, the minimum enclosing circle is determined with its centre. This would be the centre of the palm.

FingerTip Detection:

The finger tips were also detected to ensure that the output of the sign language interpreter is much more reliable. The fingertip was determined by iterating through three points at a time on the contour. The three points on the contour must be separated from each other by at least a distance of 20 points between them. With this method, all the set of three points on the contour that have a cosine value of the angle between them to be less than a threshold [that was set 0.6] was determined. The points obtained after the above method of filtering will also have false fingertips. These false fingertips can be removed by using the following algorithm. This is done by calculating the angle formed by the centre of the minimum bounding rectangle and first and second point of the set of three points. If this angle is less than 90 degrees, then the middle point of these three points is considered as the fingertip. Else if the angle is greater, this set of points is ignored and the next set is determined. After iterating through all the points, the fingertips can be determined.

Sign Detection:

To implement the sign detection we had tried multiple approaches out of which we found the last method more reliable. The different approaches tried are:

1) Template Matching - This method doesn't work unless you have the fixed template for matching the image.
2) Feature or Pattern Matching - This method matches all the features of the image with the other image. So we extracted a region of interest on both the images and tried to do a match. But deciding on the number of pattern match threshold value was difficult because while trying with different images, the number of features matched kept changing. So this method also was not reliable.
3) Determining which finger is raised - In this approach, the user would have to first place his hand in front of the camera for some time for a calibration sequence. In this sequence, the centre of palm, fingertips, and the distance between each fingertip and the centre would be determined. But just the

finger length wouldn't be enough to determine which finger has been raised. So another property was needed, and hence we decided to determine the angles formed by the fingers with a constant reference. We tried multiple reference lines which were used to determine the angles, but none of them were reliable enough to give the constant angle for different orientations of the hand. So we had to drop this plan too.

4) Match Shapes - In this approach, we would maintain a database of images for different signs in different positions so that when this database of images was matched with the hand shown by the user, it would provide a reliable result. But the match shapes gives a good match value even if the difference between the two images is just finger. So to make this comparison with the database easier and more reliable, we first determined the number of fingertips that have been raised. Keeping this in mind, we had classified the database of images into multiple folders based on the fingers raised. For example, if the number of fingertips detected were three, we would compare with the database of images which have three fingertips, i.e., the numbers 3, 6, 7, 8, 9. This approach would reduce the amount of time taken to match and also the give a more reliable match value. Based on multiple tries, we found that a threshold for match value of 0.15 to be more reliable [If the match value is close to zero, that means that a perfect match is obtained]. So to improve reliability we also do a check such that, the output is displayed only if 15 consecutive frames gave the same match result. This provided a very reliable output even though the performance had to be compromised a little.

Text to Speech Conversion:

The text to speech conversion was implemented by using a library by python named **pyttsx**. This library has API's which can be used for converting text to speech. The only drawback with this library is that the only function that the library provides for speaking out the text is a function called runAndWait(). This function converts the text to speech, but waits until the text is spoken. This adds a delay to the execution of the code which eventually reflects on the output by making the frames not continuous and have a jitter.

Future Expansions:

1) Optimize the code so that the execution time is as small as possible. Determine means in which code can be parallelized.
2) Implement the code in hardware [Altera De1-SoC or BeagleBone Black] and find the execution time for the code.
3) Find much more reliable means of identifying which finger is raised and if possible identify the joints in the hand.
4) Increase the database by taking the images of multiple gestures in different positions to give a much more reliable output.

References:

1) http://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html#gsc.tab=0
2) A thesis document that we found online. This has been attached in the zip file submitted.
3) http://pyttsx.readthedocs.io/en/latest/
4) https://www.youtube.com/watch?v=Z78zbnLlPUA&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq