Senior Design I
CSC 59866
Prof. Camp & Prof. Wei

Project Specification:

American Sign Language Translator using Gesture Recognition

Fall 2016

By:
Eun-Jung Park
Namita Sehdev
Roger Frogoso
OVERVIEW

1. Introduction
   1.1. Motivation and Significance of our Project
       - Cultural Significance of Sign Language
       - Utility of Proposed Project
   1.2. Existing Programs

2. Background
   2.1. Informational Background
       2.1.1. American Sign Language
           1) Numbers
           2) Alphabets
   2.2. Technical Background
       2.2.1. Gesture Recognition Algorithms
           1) 3D Model & Skeletal Model
           2) Features
       2.2.2. Tools for Image Processing and Machine Learning

3. Technical Description
   3.1. Project Specifications
       - Translate numbers and alphabets
   3.2. Forward Processing: Sign Language to Text & Speech
       3.2.1. Gesture Detection
       3.2.2. Gesture Interpretation
       3.2.3. Text to Speech Conversion (TTS)
   3.3. Reverse Processing: Text & Speech to Sign Language
       3.3.1. Speech to Text Conversion (STT)
       3.3.2. Text to American Sign Language (ASL)
       3.3.3. Gesture Simulation

4. Conclusion

5. Reference
1. INTRODUCTION

Our motivation for the project is to allow people who are deaf or dumb to have normal conversations with others through a program that inputs sign language in text form to allow the other person to understand. This process will go vise versa. Our goal is to provide basic program that translates numbers and alphabets into American Sign Language. If our program is elaborated after our project, it can be a program that allows people to have free conversation whether a person is disabled or not. This kind of program is significant for it can be even more elaborated to a program that translate conversations into different languages as well. This way each individual can converse in their native language giving them the homely feel.

Our approach to this project is start off small and progress gradually. We will first find an algorithm that can do the translation both ways. Then we will learn the American Sign Language for numbers and then learn the letters of the alphabet and finally translate certain words. As everyone knows that some words such as “baby” and “bad” are some of the words in sign language that require symbols rather than words. For these kind of terms we will make videos that will be able to generate these actions with the use of a camera to text instance. Then we will create a reverse translation where people who do not know sign language can say their response into the program that will either give the response by text or a sign language video. Finally, we will implement the features of the program and how it would look to the naked eye.

After researching, we noticed that there are many existing apps such as the “Prodeaf Translator” on android's play store where people can input a sent or a phrase and there is a virtual robot that translate it to sign language with the hand gestures. Since androids and apple apps differ, another similar app in the App Store for Iphones “ASL translator” where there are videos for the deaf person to watch when a text is inputted, We will keep these types of differences in mind when building our app since we would like to make it more versatile and different than other apps.
2. BACKGROUND

2.1. Informational Background

2.1.1. American Sign Language

In the United States and some parts of Canada, majority of the deaf communities use American Sign Language (ASL) as their main method of communication. It is a visual language: meaning that it is a language that is expressed through gestures made mainly by the hands and face. ASL carries just as much information as any other language through its complexity. It has its own grammar, culture, terminology, and it even has different dialects. Because of its complexity, it will prove to be a great challenge to create a program that can fully translate ASL.

1) Numbers

![Sign Language Numbers Image]

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
2. Technical Background

2.1. Gesture Recognition Algorithms

The main goal of the gesture registration is to come up with mathematical algorithms that will allow the program to interpret human gestures. We have come up with an approach that involves 3 different types of algorithms that will resonate with each part of the program. There are algorithms for the 3D model, skeletal model as well as the physical features of the program. Each approach uses a different type of data input.

1) 3D Model & Skeletal Model

In this part of the program we will use an algorithm that is used to create a 3D or three dimensional model which detects an object and its position within a certain radius of the camera to identify the gesture portrayed. The detection is the most important part of the program since a
gesture can only be translated to text if a gesture is registered into the program. These models can be created in numerous ways: from a high complexity approach by using polygon meshes, to a simpler mapping method by using primitive shapes such as cylinders and circles. This approach, however, is somewhat inconvenient because it requires an excessive amount of computing power and lacks systems that can perform real time analyses. As far as the skeletal model is concerned it is a spatial model that has an algorithm which uses a simpler set of parameters such as joint angles and segment lengths. In this part of the project we will provide a skeletal frame that can be analyzed by using their position, orientation, and relationship with other joints and segments. Because this approach uses a significantly less amount of parameters, the algorithm runs faster, and uses less resources than the 3D model-based algorithm.

2) Features

The features of this program will be user friendly. The model will be derive all the gestures by an image or a video. In the first part we will use template databases to approximate the outline of the object. The models used to represent this program are known to be used in hand tracking. From this hand tracking technique, the program will analyze a person’s skin color from dark to light so that it can recognize different races when coming in contact with the camera in the program. Several color spaces have been proposed including RGB. There will be color spaces efficiently separating the chromaticity from the luminance components of color. All shadows and lighting changes will be taken into consideration. The distribution of colors will be achieved by a form of a single Gaussian distribution or a mixture of different gaussian equations. To identify skin color, the program will estimate the correct skin shade from the parameters of the probability density functions. Second, the program will recognize the shape and size of a person’s hand. Since each individual has different hand sizes, the user’s hand is directly extracted as a contour by assuming a uniform background and performing real-time edge detection in this image. It will detect from the arm first since it is more visible to the camera then the palm and finally the fingers. A hypotheses of hand 3D models are evaluated by first synthesizing the edge image of a 3D model and comparing it against the acquired edge image. Finally we will use the process of boosting. Boosting is a general method that can be used for improving the accuracy of a given learning algorithm. Accuracy is the shown by the help of gesture templates.

2.2.2. Tools for Image Processing and Machine Learning

In our project, we will need some tools to do image processing for gesture recognition and machine learning for gesture interpretation.

1) Image Processing
OpenCV is a library with lots of algorithms related to Computer Vision and Machine Learning. It supports many programming languages like C++, Python, Java etc. OpenCV is an appropriate tool for fast prototyping of computer vision problems. So, we will use OpenCV as our image processing tool.

2) Machine Learning

Scikit-learn is a machine learning library for the Python. It has various machine learning algorithms for classification, regression, and clustering. Once we get the preprocessed images from image processing, the data will be refined again by clustering so that it can be categorized into each sign language with the similar hand gesture. Then, the clustered gestures will be used as a train set for the gesture that corresponds to each sign language. Using the trained model, we will test the input that we get from a camera.

Scikit-learn will be used for our program to understand and analyze the meaning of a hand gesture and result the output into text. As an example, the sign language for the alphabet ‘A’ should be captured despite its background. The sign language for ‘A’ will be distinct from other sign languages - ‘E’, ‘M’, ‘N’, ‘S’, and ‘T’ - even though their hand shapes are similar to each other. As the result, it should be translated into text ‘A’.
3. TECHNICAL DESCRIPTION

3.1. Project Specification

We are proposing to make a program that has two main functions; forward processing and reverse processing. Forward processing is to translate American Sign Language into English text and speech whereas reverse processing is to translate English speech into American Sign Language. For the former function, our program will take a live video input and return an audio output. For the latter, our program will take an audio input and return a video output. Because of the complexity of ASL, we will limit our program to translating numbers, letters, and several common words and phrases.

3.2. Forward Processing: Sign Language to Text & Speech

As shown in our diagram, this process can be divided into three major categories: gesture detection, gesture interpretation, and text to speech conversion.

3.2.1. Gesture Detection
Because ASL heavily relies on hand gestures, the first step in our approach to recognize our target gestures is to detect the hands. Our goal in this section is to discriminate and isolate the hands from the rest of the background. This is a necessary step because it simplifies the data that will be used for interpreting the gesture. In order to discriminate and isolate the hands from the image, we will use image processing techniques to derive our target visual features; we will most likely use some of the following features: skin color, silhouettes, and contours. (Zabulis et al.)

Skin color is a convenient feature because it is usually distinct in hue-saturation space, and is not greatly affected by changes in lighting. Techniques that differentiate skin color from its surroundings usually rely on histogram matching or tables that consist of training data for the skin. One drawback of this feature, however, is the variability of skin color in different lighting conditions, which can lead to failures in, or falsely detecting skin color; light skin colors may be harder to detect in bright lighting conditions, while dark skin colors might be too similar to their background in dim lighting conditions. (Pavlovic et al.)

Silhouettes are another parameter of interest because it is very simple and easy to extract. The only drawback with this approach is the risk of losing information. As shown in our background, the gestures for the letters “N” “S” and “T” are all very similar and only differ in the placement of the thumb. If the

---

1 https://ilab.cs.ucsb.edu/index.php/component/content/article/12/31
silhouette of the hand was taken it might not be possible to determine which letter is being presented. Contours are similar to silhouettes but focus on extracting the edges on an image. Some methods derive the edges from silhouettes, making them equivalent; however, edge detection techniques can also be applied to colored and gray-scale images.

To perform these image processing techniques, we will be using OpenCV. This library supports Numpy, which is an optimized library for numerical operations in Python. That means openCV arrays can be converted into Numpy arrays to increase the speed of numerical operations on large sets of numbers much faster. We will start using OpenCV of Python version for its simplicity and code readability. However, we can switch to C++ version later if the speed of Python version is not good enough for in real-time recognition of hand gesture. Python can be easily extended with C++ to make our program as fast as original C++ codes. OpenCV also contains statistical machine learning functions for k-nearest neighbor algorithm, naive Bayes classifier, artificial neural networks, Support Vector Machine, etc. On top of that, OpenCV can be used with other machine learning libraries like SciPy and Matplotlib which supports Numpy.

3.2.2. Gesture Interpretation

Interpreting gestures can be broken down into two levels of complexity: posture and gesture interpretation. Posture interpretation is the simpler case because it only deals with the interpretation of non-moving figures. On the other hand, gesture interpretation adds another level of complexity by analyzing the movement of figures. In ASL, most of the hand gestures for numbers and letters do not involve any motion after the sign is formed; this will allow us to use posture interpretation approaches. Because our target vocabulary is relatively small, we can extend this approach to gesture interpretation by treating it as a series of hand postures (Zabulis et al).

Depending on the type of data we extract from our detection method, there are a couple of techniques that can be used to interpret a hand’s posture. One technique, called template matching, performs an algorithm that compares an input image pixel by pixel to a template image. A score will be given based on a set of predetermined measurements of similarity. For an image to match a template, a certain threshold score must be reached, and the template with the highest score will be selected. One of the challenges this technique has is the problem of image scaling and rotation. Attempts to solve this problem range from making normalization functions for scaling and rotations, to using multiple views for each template image. (Zabulis et al.)

Contour and silhouette matching are conceptually similar to template matching; they compare an input contour or silhouette to template images. Because contours and silhouettes are
very simple, the measures for similarity are different from template matching; similarities in edges and continuity are the focus instead of factors like intensity. (Zabulis et al.)

Although we have not decided our interpretation approach, it is important to understand the machine learning nature of gesture interpretation algorithms. In essence, machine learning techniques use mathematical models of observable phenomenon to provide predictions for unobserved data (Belgioioso et al.) For our case, the phenomenon being represented as mathematical models are the hand gestures and their corresponding meaning, and the unobserved data is the real time detection of a person performing ASL. We can also see that we are dealing with a classification type problem and can take inconsideration some machine learning classification techniques. To name a few, we have the k-Nearest Neighbors, Discriminant Analysis, Support Vector Machines, and Relevance Vector machines. Scikit-learn is a library that contains some algorithms relating to these techniques, which is why we will use it for our project.

3.2.3. Text to Speech Conversion (TTS)

Text to speech conversion is often called TTS which is an abbreviation of Text-To-Speech. Festival (the Festival Speech Synthesis System) is a framework for TTS. It builds speech synthesis systems with various example modules and offers a full text to speech system. Since it is written in C++, we will use it through the Python wrapper Pyfestival. In order to keep our focus on gesture recognition and to reduce development time, we will use this existing library. In our project, Pyfestival will be used to convert text into English speech. The text input for this tool will be obtained from the gesture interpretation.

3.3. Reverse Processing: Text & Speech to Sign Language

Like forward processing, reverse process is also consist of three major categories; speech to text conversion, text to ASL translation, and gesture simulation.

3.3.1. Speech to Text Conversion (STT)

Reverse process starts starts from where the forward process ended. That is speech to text conversion. It is often called STT in a short form for Speech-To-Text. One of tools for STT is CMU Sphinx. It is a group of speech recognition systems developed at Carnegie Mellon University. It includes series of speech recognizers and acoustic model trainer. Among the four different versions of Sphinx, we will use Sphinx 2 for it is a semi-continuous speech recognition.
system that focuses on real-time recognition of spoken language. Pypi is a Python library for speech recognition that supports several speech recognition engines and APIs including CMU Sphinx. This tool will be used in our program to convert spoken English that we acquire from microphone into written English.

3.3.2. Text to American Sign Language (ASL)

We will use the stored model that we get from training process of gesture interpretation. The text value retrieved from spoken English will be matched to the corresponding ASL model. Then the ASL posture or gesture will be passed to the next stage to be simulated on the computer screen.

3.3.3. Gesture Simulation

On the final stage of reverse processing, we will visualize the sign language that we get from text evaluation to the computer screen using computer graphics. We chose VPython as our graphic tool for it is a Python library with simple but good enough modules to simulate hand gesture. Using functions on VPython, we will create simple 3D graphic modules like spheres and cylinders and arrange them so that it can be understood as a hand. Then, we will use those simple hand model to simulate the sign language on the screen.

The interpreted ASL input can be a static sign language like ‘A’. For this case, the input is considered as posture rather than gesture as it was in the gesture interpretation. These static posture input will be simulated with the corresponding static hand model. On the other hand, the ASL input can be a sequence of static sign language like ‘ABC’ or a dynamic sign language like ‘Z’. For this case, the ASL input is a gesture rather than a posture, so an animated hand model will be used to simulate the corresponding motion of ASL. For example, upper and lower arms are going to be represented as large cylinders. Then the hand motion can be simulated with the motion of smaller cylinders representing fingers.

We will make the graphics as simple as possible since computer graphic is not our goal in this project. However, our graphic will be good enough to differentiate the sign languages in our data boundary which includes only alphabet, number, and several simple words.
4. CONCLUSION

Our program will be delivered as a runnable software on computer. It will recognize American Sign Language meaning numbers or alphabets from camera input, then it will output the equivalent number or alphabet in spoken English or written English. The program will also be able to do the translation in the opposite direction. If it takes the spoken English or text as an input, it will display the corresponding the American Sign Language on the screen.

According to a paper done by Brault of U.S. Census Bureau\(^2\), the estimated deaf population in the United States is 29,572,076. The record is based on 2008 American Community Survey, and deaf population takes up about 1% of american population in 2008. The number will be larger if we add people who cannot speak into the consideration.

For the people who can not hear or who can not speak, our program will remove the barrier that comes from their disability in terms of numbers and alphabets. We think it is important and meaningful start because alphabet can express any words, and words with numbers are basic element of communication.

For the other people who wants to communicate with the deaf or dumb, our program will save time and effort they spend to study American Sign Language. Unless they have previous knowledge about it, it is not easy for normal people to communicate with the deaf or dumb naturally. If they were to learn American Sign Language, it takes lots of time. Our program will saves their time by letting them skip the effort they pour on studying for numbers and alphabets in American Sign Language.

Most of all, our program will be a foundation stone for a translator between American Sign Language and spoken English in both direction. If the data set is extended later beyond the numbers and alphabets, the upgraded program will reduce the time and effort of normal people significantly and increase freedom of conversation for the deaf and dumb dramatically.

5. REFERENCE


2) X. Zabulis, H. Baltzakis, A. Argyros, “Vision-based Hand Gesture Recognition for Human-Computer Interaction”, Institute of Computer Science Foundation for Research and Technology - Hellas (FORTH) Heraklion, Crete, Greece, \{zabulis,xmpalt,argyros\}@ics.forth.gr

3) Giuseppe Belgioioso, Angelo Cenedese, Giuseppe Ilario Cirillo, Francesco Fraccaroli, Gian Antonio Susto, “A Machine Learning based Approach for Gesture Recognition from Inertial Measurements”