University of Arkansas – CSCE Department
Capstone I – Final Proposal – Fall 2020

Sign Language Teacher/Interpreter using Xbox Kinect

David Clairmont, Johnny Doan, Jack Gaither, Nick Hester, Sam Witucki

Abstract

The problem that we have discovered is that learning American Sign Language (ASL) is difficult, as it is with learning any language. Many different platforms help with teaching ASL, but through experience, online sign language teachers lack the ability to keep the student fully engaged and interested in the subject. Our objective is to develop a sign language teacher to improve the learning process by actually engaging with the student, with hopes of maintaining the desire to learn the language to its fullest extent. Overall, our objective is to make learning ASL easier and more fun.

Our approach is to train an AI with an Xbox Kinect by having a reference sign each letter or a simple phrase. The significance of the sign language teacher is that it would help teach people how to better communicate with deaf/hard of hearing individuals. It would allow a person to expand their knowledge of the language and give them a special benefit in most situations.

1.0 Problem

The problem that many people face when they start to learn sign language is that it is difficult to do, and even more difficult to do correctly. Learning sign language is unique because it's a completely visual language. This means that to practice sign language, you must either sign with someone else who knows sign language or record yourself signing and replay it.

In the U.S. alone, around 250,000 – 500,000 people rely on sign language for their day-to-day communication because of their hearing disability [12]. When you factor in the number of people who then rely on sign language to communicate with people with a hearing disability, the number grows even larger. An application that provides a feasible and interactive way to learn sign language would allow an entire group of the U.S. population to effectively communicate in public settings.

Without this application, sign language would remain a very difficult language to learn individually. To accurately learn sign language without forming bad habits, you would most likely have to join a class where a sign language educator could differentiate correct and incorrect gestures and provide feedback. Without something like this application to interpret sign language, people who rely on it will inherently have a more difficult time doing everyday things that are specifically catered to people who do not have a hearing disability (i.e., audio-only drive-through option at restaurants).

2.0 Objective

The objective of this project is to write software capable of correctly interpreting American Sign Language (ASL) using Xbox Kinect hardware.

3.0 Background

3.1 Key Concepts

Digital image processing is using a computer to analyze an image using algorithms [1][2]. This kind of processing can range from compression to enhancement and restoration. Some special image processing will likely be necessary to correctly pick out human bodies and their structure from the live image feeds.

Once an image has been processed, we will have to use computer vision concepts to correctly interpret the image. Computer vision is a broad field that deals with how to get computers to interpret images the way humans do [3][4]. This would be considered a high-level understanding of the image. Various algorithms are used to try to achieve the broad range of image interpretation that humans currently possess. Often these algorithms are complicated and require AI to function correctly.

AI Action recognition is the process by which an AI can recognize a specific action taking place in an image. It’s also considered a subset of computer vision. Usually, the actions the computer is trying to recognize are different human actions, like walking, running, talking, etc. Often this requires deep learning to work effectively along with a large amount of input data from many contexts to train the AI [5].

Once the computer vision process has been completed for a given image, it must be compared to a library of sign language gestures. We’ll be using American Sign Language specifically, although there are other sign languages [7]. It has unique signs for letters, numbers, and words all of which are formed using the hands, face, and other body language. While it is commonly used in English speaking countries, it has its own grammatical rules different from English grammar [6].

3.2 Related Work

Dr. Lora Streeter worked on a very similar project for her dissertation [8]. She used Kinect along with several different algorithms to try to identify user gestures and translate them into instructions for a gesture-based programming language. Early in the paper, she mentions that an application of her project could be helping people with certain disabilities, even mentioning a theoretical program that could recognize sign language gestures, which is a major part of what this project is trying to achieve. Our project will involve building a system that will likely interpret images similar to how hers did, but instead of using it for programming, we will use it for sign language.

We were able to find a project on GitHub created by Harsh Gupta, Siddharth Oza, Ashish Sharma, and Manish Shukla that is essentially a working implementation of what this project hopes to achieve. They created a sign language interpreter using deep learning in a day for the

University of North Texas's 2019 Hackathon with Python and several other packages. The interpreter was capable of recognizing 44 different signs with high accuracy [9]. Our implementation will probably be similar, but we aim to have it recognize a larger number of signs.

Another similar project is the Kinect Sign Language project created by Svilen Popov on GitHub. It's a sign language translator that uses Kinect to get user input [10]. The project appears to be completed and working, but the description doesn't explicitly state that it works. Whether it's working or not, a Kinect-based sign language interpreter would be very useful to work off of for our project.

PyKinect2, a project that enables its users to create applications for Kinect using Python [11], is more general than the others listed but will be helpful. Having a Python library build specifically for Kinect will help us make progress much faster than if we didn't have it.

4.0 Design

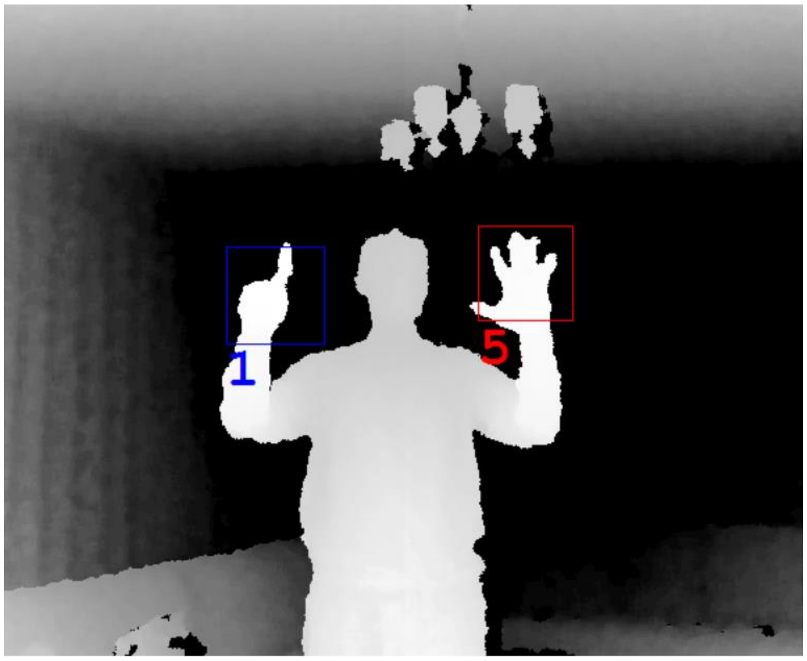
4.1 Requirements, Use Cases, and Design Goals

1. **Requirements**
    - 1.1. Connect to Kinect API through Python
    - 1.2. Able to convert Kinect sensory data into Numpy datasets
    - 1.3. Must allow for extensive training of datasets in order to improve AI accuracy
    - 1.4. Must be able to predefine different sign language letters, words, and phrases into identifiable data for AI
    - 1.5. Allow for a user to sign to the Kinect and receive back the identified sign
2. **Use Cases**
    - 2.1. User runs the application, allowing access to the Kinect camera
    - 2.2. User can provide a sign and receive the correct answer
3. **Design Goals**
    - 3.1. Fast, efficient code
    - 3.2. Easy access for modifying set data and loading trained model

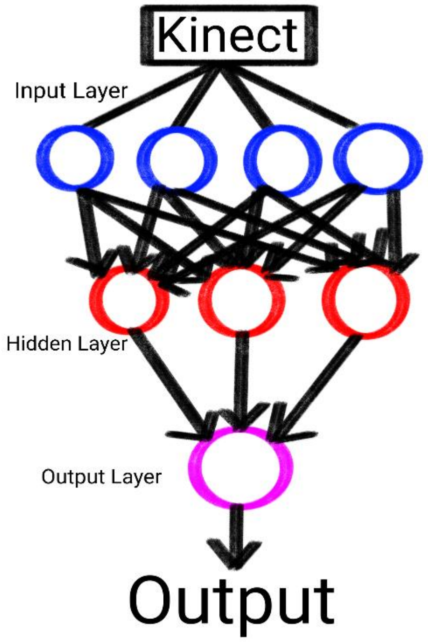
4.2 High-Level Architecture

To implement the sign language recognizer, we would start by creating a Python file that is able to connect to the Kinect API. By connecting to the Kinect API, we can access features such as Kinect body tracking and coordinate mapping. This will allow us to capture the left and right-hand joint coordinates from the device. From here, we will be able to convert the data into usable inputs for the neural network. At this point, we will have to implement the neural network using Python libraries such as TensorFlow and Numpy. The network will be trained with the relevant signs made in different positions and distances.

The picture shows how the Kinect can be used to identify different hand symbols [13]:



Live video feed is received from the Kinect and converted into input for the input layer. The input goes through hidden layers to distinguish identifiable traits and determine which possible alphabet sign it might be. The final layer is the output layer, where all possible outputs will reside. Upon receiving output in the output layer, a result will be sent back to the program representing the neural network's determined alphabet letter. Below shows a diagram of how the data gets processed:



4.3 Risks

Risk	Risk Reduction
Inaccurate Sign Language	Extensive research of ASL
Inaccurate Neural Network	Train with many datasets in order to improve accuracy
Slow code	Write out algorithms and optimize for the best complexity time

4.4 Tasks – This is a tentative list of tasks for the Sign Language Interpreter. It is broken down into phases based on engineering design principles: Preparation, Design, Implementation, Testing, and Documentation Phases. Below is the list of tasks:

1. Preparation Phase

1.1 Understand how to program the Kinect. Based on this, we will determine what software(s) are necessary to capture the person's motion. To have a general basis for our approach, we will reference some of the related works mentioned before.

1.2 If needed, install the necessary software(s) and obtain our equipment, which consists of the Xbox Kinect and a device that connects to the Kinect. Once we have our materials, we can test the equipment to ensure everything is working properly before diving into the project.

1.2.1 Testing will include checking if the Kinect is compatible with our device and that the Kinect itself can capture a person's motion. If there are issues, we will need to report them and possibly get a replacement or use a different software program.

2. Design Phase

2.1 Create an application that allows the user to use the Kinect.

2.1.1 This application will work with the Kinect API. Depending on how we progress into the project, the application will work with other sign categories (i.e., numbers, time, people, questions, etc.).

2.2 Develop a library/database that stores the ASL words/letters that have been programmed.

3. Implementation Phase

3.1 Implement the first three letters of the alphabet (A, B, C). This will serve as the foundation of our implementation for the remaining letters.

3.2 Complete the remaining letters of the alphabet. The alphabet will either be split to each team member or be split to 2-3 members while the remaining will work on the application. With every member working on a set of letters, it ensures that everyone understands how to program the Kinect. A sample distribution is shown below:

Team Member	Set of Alphabet Letters
David Clairmont	D – H
Johnny Doan	I – M

Jack Gaither	N – R
Nick Hester	S – V
Sam Witucki	W – Z

3.3 Once we complete the alphabet, we can then extend to other categories of sign language (i.e., numbers, time, people, questions, etc.) if we have enough time.

4. Testing Phase

4.1 Run the application and check if it fulfills our objective and expectations.

4.1.1 Ensure that the application is what we anticipated while user-friendly.

4.2 Test to see how accurate the Kinect can depict each letter of the alphabet.

4.2.1 Testing will include different angles of the person's hand with respect to the Kinect and different distances from the person to the Kinect.

4.2.2 Testing requires being physically in front of the Kinect. Due to COVID-19 and for safety precautions, it's best if one member holds all the equipment and tests out each letter. Alternatively, we can see if the Kinect can pick up video recordings of each member signing (subject to change if there's a better solution).

5. Documentation Phase

5.1 Report the results of our application. Screenshots and/or videos of the walkthrough of the application will be provided so the user understands how it works.

5.2 Record a demonstration of the Kinect being able to read the person signing.

4.5 Schedule – Below indicates our proposed timeline for each task listed in section 4.4:

Tasks	Dates
1. Equipment/Software Preparation.	11/16-11/30
2. Test the Equipment. If there are issues, report and get necessary replacements.	11/30-12/12
3. Design a Python application for the Kinect.	1/11-1/25
4. Develop the library/database that holds all the ASL words/letters that have been programmed.	1/25-2/1
5. Implement the first 3 letters of the alphabet and test the accuracy. Add the letters to the library.	2/1-2/8
6. Test the application, ensuring that the library is properly connected and running properly. Make necessary adjustments if needed.	2/8-2/15
7. Implement each set of alphabet letters (refer to section 4.4, step 3.1 to see the distribution of letters among team members). Add letters to the library and test if the letters work properly.	2/15-3/1

8. Make necessary fixes if the letters aren't working correctly. If there are minimal to no fixes, then we take this time to brainstorm other terms to include.	3/1-3/8
9. Implement ASL numbers. Add them to the library and test if the letters work properly. Make necessary adjustments if needed.	3/8-3/22
10. Implement ASL terms relating to time. Add them to the library and test if the letters work properly. Make necessary adjustments if needed.	3/22-3/29
11. Implement ASL terms relating to people**.** Add them to the library and test if the letters work properly. Make necessary adjustments if needed.	3/29-4/5
12. Record and report the necessary information for the final document.	4/5-4/19
13. Make any final adjustments before submitting the project.	4/19-4/30

4.6 Deliverables

- Design Document: Contains a listing of each major hardware and software component. This can also include how to use the Kinect when signing and videos showing how the Kinect reads the person's body.
- Database scheme and initial data: The library/database with all of the programmed letters/words stored in it.
- Web site code: The code for the web site.
- Python code for training the AI, gathering and storing optical data, digital image processing, etc.
- Final Report

5.0 Key Personnel

David Clairmont – Clairmont is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses in Programming Foundations I & II, Programming Paradigms, Software Engineering, and Database Management Systems. He is currently an undergraduate research assistant for Dr. Qinghua Li. Clairmont will be responsible for creating the Python files to connect to the Kinect API and programming some letters of the alphabet.

Johnny Doan – Doan is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses in Programming Foundations I & II, Programming Paradigms, Database Management Systems, Software Engineering, and Algorithms. Doan will utilize his leadership experience from student organizations to keep track of each member's progress. He will also use his class experiences to work on the set of alphabets and the application that connects the program to the Kinect.

Jack Gaither – Gaither is a senior Computer Science major in the Computer Science and Department at the University of Arkansas. He has completed relevant courses in Programming Foundations I & II, Programming Paradigms, Database Management, Software Engineering, Cryptography. He has experience with running various HPC applications, image processing, action recognition/segmentation, and teaching python tutorials. Gaither has also interned for the past three years with the Texas Advanced Computing Center, starting as an REU student and then mentoring for the past two years for various student programs. He now works with Dr. Luu in the CVIU lab on campus doing action recognition/segmentation. He will be responsible for working on the set of alphabets and testing each of the letters.

Nick Hester – Hester is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses which include Programming Foundations I & II, Programming Paradigms, Database Management Systems, and Software Engineering. He has experience creating a website that will come in handy when creating the website for this class. Hester will be the main reference for the initial signing training. Hester is capable of signing the full alphabet along with the simple phrases that will be used to train the AI and will work on the testing portion of the project along with programming the set of alphabets.

Sam Witucki – Witucki is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I & II, Programming Paradigms, Software Engineering, and Database Management Systems. He is currently employed as an IT/Engineering Intern, writing software at J.B. Hunt. He will be responsible for writing some of the Python code for this project as well as providing an extra reference for the sign language interpreter.

6.0 Facilities and Equipment

For this project, the main equipment we're using is the Xbox Kinect. This will act as our main medium for our image capture. We will most likely utilize a GPU machine that can be accessed via SSH for our initial image processing and model training.

7.0 References

[1] Digital image processing, https://en.wikipedia.org/wiki/Digital_image_processing

[2] Digital Image Processing Introduction, https://www.tutorialspoint.com/dip/image_processing_introduction.htm

[3] Computer vision, https://en.wikipedia.org/wiki/Computer_vision

[4] What Is Computer Vision?, https://www.pcmag.com/news/what-is-computer-vision

[5] Introduction to action recognition, https://www.coursera.org/lecture/deep-learning-in-computer-vision/introduction-to-action-recognition-Ift3j

[6] American Sign Language, https://www.nidcd.nih.gov/health/american-sign-language

[7] American Sign Language (Wikipedia), https://en.wikipedia.org/wiki/American_Sign_Language

[8] Streeter, L., "Teaching Introductory Programming Concepts through a Gesture-Based Interface" (2019). *Theses and Dissertations*. 3240. https://scholarworks.uark.edu/etd/3240

[9] Sign Language Interpreter using Deep Learning, https://github.com/harshbg/Sign-Language-Interpreter-using-Deep-Learning

[10] Kinect Sign Language, https://github.com/sgpopov/kinect-sign-language

[11] PyKinect2, https://github.com/Kinect/PyKinect2

[12] Mitchell, R., “How Many People Use ASL in the United States? Why Estimates Need UPdating,” 2005. https://www.gallaudet.edu/documents/Research-Support-and-International-Affairs/ASL_Users.pdf

[13] Mustafa, E., Dimopoulos, K., “Sign Language Recognition using Kinect,” 2014. https://www.researchgate.net/profile/Konstantinos_Dimopoulos2/publication/266144236_Sign_Language_Recognition_using_Kinect/links/5447bc8b0cf2f14fb81228ca/Sign-Language-Recognition-using-Kinect.pdf