Gesture Glove

Authors: Sophia Lau, Rachel Tang, Stephanie Zhang: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—This paper explores the use of sensor gloves in sign language detection. This project uses a glove embedded with sensors to recognize and translate ASL letters. This glove aims to generalize to all users and to minimize physical components for satisfying user experience.**

***Index Terms*— American Sign Language (ASL), ASL recognition, Fingerspelling, Flex Sensors, IMUs**

1 INTRODUCTION

The motivation behind our project is to help people who sign in American Sign Language (ASL) communicate with those who do not understand it. Thus, our target users are people who use ASL as their primary form of communication, such as those who are hard of hearing or are not able to speak. To achieve this goal, we are creating a system containing a glove and a computer that will recognize gestures and display them as audio output through speakers. From a user standpoint, we would ideally have a gesture classification accuracy of 100% and an undetectable latency (less than 15 ms).

We have chosen to create a system using sensors as opposed to taking a computer vision approach. An advantage of using sensors over computer vision is that sensors allow the product to be more portable since it would not require the user to have a camera facing them at all times in order to detect gestures. Additionally, sensors are less variant to external factors such as lighting. We will also use one of Perceptron, Support Vector Machine (SVM), Neural Network, K Nearest Neighbors (KNN), and Random Forest classifiers to determine what is being signed by the user.

2 DESIGN REQUIREMENTS

As the Gesture Glove will be a worn device, it's crucial to provide a comfortable user experience and accurate sign recognition. Based on the user requirements we have previously mentioned in the introduction, we have established the following design specifications regarding the product's accuracy, latency, gesture frequency, and craftsmanship.

The Gesture Glove should have a classification accuracy of at least 90%. This accuracy requirement means that whenever the user makes an ASL sign, the classification model should output the correct letter corresponding to the sign 90% of the time. Ideally, we want to achieve 100% accuracy to satisfy the user requirements and to ensure top-quality experience. While it may be quite difficult to reach such an accuracy with any type of classification models, having a recognition rate of 90% or more will not greatly affect user experience. This requirement is based off the average typing accuracy of roughly 90% [**1**]. Since typing and signing are common forms of communication, it's reasonable to have approximately the same accuracy. We aim to reach as high of an accuracy as possible with 90% simply as a baseline.

In terms of testing, we will be performing accuracy tests with various users and making adjustments to the classification model as needed if the product does not reach a satisfying recognition rate. Along with accuracy tests, we will perform an experience survey to see user satisfactions with our product's accuracy.

The Gesture Glove should have a latency of at most 100ms. Besides having a high accuracy, the Gesture Glove should also output the results of the sign recognition in a short amount of time. Our goal aligns with the user's requirement: to have an undetectable latency for the Gesture Glove. To determine an ideal latency, we start by looking through various research papers. We find that while users can detect latency as low as 33ms, the user experience do not suffer significant losses until latency is over 100ms[**2**]. Latencies over 100ms are much more noticeable to the users and significantly degrade user experience. We have determined that having a latency of less than 100ms is ideal. To ensure we achieve this latency, we will set up timers in our code to check for how long the communications will take between the glove sensors and the computer and how long the classification model takes to recognize a sign.

The Gesture Glove should recognize two signs per second. Upon researching the rate of ASL signing, we find that mean signs per second is around 2.5 signs, not taking into account of pauses in between signs[**3**]. Considering this information, we have determined that an average person likely makes two signs per second, including pauses between signs. Since we want the users to feel natural when signing (not too fast and not too slow), we decide to use the average signing rate as a baseline for the frequency.

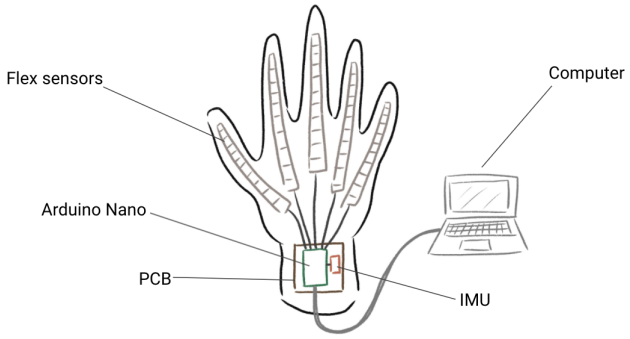
The Gesture Glove should have good craftsmanship, Our product will be a wearable device and its weight and stiffness should not affect the user experience, especially in making signs. After the glove is fabricated, the total weight of the glove should not exceed 200 grams so that the users can feel comfortable while wearing. The completed glove should also be flexible. Bending of fingers should be easily achieved without users feeling impeded when signing.

We will be conducting user experience surveys during data collection to verify user satisfaction. If the users do not feel comfortable using the glove or had trouble signing with it, we will collect feedback to make further improvements.

3 ARCHITECTURE OVERVIEW

3.1 System Sketch

Our solution approach entails installing flex sensors along the length of each finger on a glove and also placing an IMU and Arduino Nano on the back of the glove near the wrist. The placement of components is visualized in Figure 1.



(a)

Figure 1: Overall system sketch

The flex sensors will collect information about the bend of each finger while the IMU will collect information about the orientation of the hand such as if the palm is facing up, down, left or right. The data from the sensors will feed into an Arduino Nano which will sample the data from the sensors and format the values. The Arduino Nano will then send the values over a serial connection to the computer which will analyze the data and run a classification algorithm to determine the pose of the hand.

3.2 Block Diagram

Our block diagram divides our system into two subsystems: the glove and the computer. As shown in Figure 2, the glove part of the system consists of the flex sensors, IMU and an Arduino Nano. The Arduino Nano will be connected to the computer with a USB cord. Through the USB connection, the computer will power the Arduino Nano which will then power the IMU and voltage divider circuits made with the flex sensors. The Arduino Nano will communicate with the IMU using I2C protocol which requires the SDA and SCL lines. The IMU will output nine data points, three measurements in the x, y, and z dimensions for acceleration, rotation and magnetic field. The Arduino Nano's analog pins will read the voltage fluctuations caused by the flex sensors in the five respective voltage divider circuits. The Arduino Nano, IMU and flex sensors will be connected with a PCB in order to keep the circuitry compact. The flex sensors will not be mounted on the PCB but will connect to some pinouts on the PCB. The Arduino Nano will run a script to continuously sample the voltage of the circuits made with the flex sensors and the values outputted by the IMU. Then the Arduino Nano will format the fourteen values (five voltages from the flex sensor circuits and nine from the IMU) into fourteen by one vectors and send them serially to the computer.

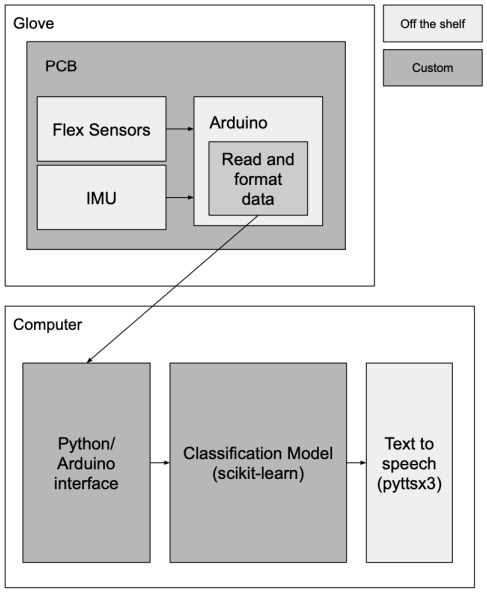


Figure 2: Overall system block diagram

The computer will run a Python script to read from the port the Arduino Nano is connected to and do any necessary parsing to manipulate the fourteen by one vectors. This parsed information will then be sent into a classification model which will determine the ASL letter being formed by the glove wearer. The computer will then speak the letter using a text to speech library. This library simply requires an input of a word or letter and will play the spoken input out of the computer speakers.

The hardware in this project consists of the flex sensors, IMU, Arduino Nano and connective circuitry. The software in this project consists of the Arduino sketch that will read, format and send the data to the computer, the Python script which will receive the data from the Arduino and the classification model which will determine the shape of the glove wearer's hand and corresponding ASL letter.

4 DESIGN TRADE STUDIES

4.1 Sensors vs. Computer Vision

The largest implementation decision we had to make was whether we would use computer vision or a combination of several sensors to identify the gestures the user is making. Both have their benefits and flaws, but we ultimately went with the sensor approach.

Computer vision has the benefit of potentially providing more data that our machine learning model can train on since the machine learning model has all the pixels in a scene (and potentially several frames) to work with to classify the gesture. Additionally, computer vision would be able to capture more minute changes in position that sensors would likely not pick up. However, with computer vision, the accuracy of our system would heavily depend on environmental factors such as lighting. Using computer vision would also make the system less portable since it would require a camera setup.

Sensors are better than computer vision several aspects. Unlike computer vision the sensor's performance is not affected by environmental lighting, allowing more consistent data collection, which would result in more accurate classification. Having sensors attached to a glove also makes our system more portable and easier to use since it would not require a camera setup and as a result, would also be a lighter system. On the other hand, sensors are slightly less sensitive to minute changes in gesture and will give similar measurements for different gestures. However, we decided that the trade-off between sensitivity to difference in poses and stability under different environmental settings was worth it.

4.2 Accuracy vs. Number of Gestures

When designing our system, we were faced with deciding how many gestures our glove would recognize. While the glove would ideally be able to identify all of ASL, it is not realistic for our system to do so. The biggest trade-off here is between the number of gestures our glove can identify and the accuracy of classification. Fewer gestures would result in a higher accuracy since the model would allow more range of motion. The machine learning model will output a class no matter what, based on the inputted data, so even if a user makes a gesture that deviates quite a bit from the "standard" pose for that gesture, it will likely still get classified correctly. Conversely, if we allow for a large amount of gestures, there would need to be minimal deviation from the "standard" pose that the glove is trained on in order for the system to classify the gesture correctly. This would both decrease classification accuracy and difficult for signers to use since everyone makes each gesture slightly differently due to difference in hand-sizes, flexibility of fingers, how they were taught, and such other factors.

While higher classification accuracy is desired, we cannot reduce the number of classes to an amount that makes the glove impractical for users. For example, a glove that can only recognize a couple gestures would not be helpful in aiding ASL users to communicate with others since communication requires much more than a couple of gestures. Eventually, we settled on having our glove classify the 26 letters of the alphabet because this allows for full communication without compromising accuracy.

4.3 Comparison of Machine Learning Models

We considered 5 different machine learning models for our system: Support Vector Machines (SVM), Nerual Networks, Perceptron, K Nearest Neighbors (KNN), and Random Forest Classifiers. Of the 5, SVMs, neural networks, and perceptrons are classifiers that we have seen past projects similar to ours use, and KNN and random forest are two other common classification algorithms in the machine learning world.

The perceptron algorithm trains the system to find a set of hyperplanes that separate the data into their respective classes. Given that our data is linearly separable, meaning that a set of hyperplanes exists to correctly classify all of our data, this algorithm would work well and be fast to use in our system since all the model has to do to classify a new data point is decide which side of the hyperplanes the new point lies on. However, due to variance in the data we will collect (since we are collecting from several people), the noise from the sensors, and the similarity in some of the gesture we are attempting to classify, our data is not likely to be linearly separable.

Support vector machines (SVM) are similar to perceptrons in that they also look for a set of hyperplanes that linearly separate the data. However, SVMs improve upon this concept by using kernel methods to transform the data into a vector space that makes them linearly separable. SVMs also finds the hyperplanes that are maximally far away from each data point in each of the classes, meaning that new data points will be more likely to fall on the correct side of the hyperplanes for accurate classification. Due to these additional methods, we expect support vector machines to perform better than perceptrons. In terms of latency for this classification method, there is the added step of kernelling the data, but that transformation should not make the latency significantly greater than that of the perceptron algorithm.

Neural networks are also similar to perceptrons since neural networks are essentially several perceptrons working in sequence. Rather than the output of a perceptron being the output of the system, it gets fed into another perceptron, and this process continues for as many layers as is specified by the model designer. Because there are several layers in a neural network, the model will be able to extract information and transform the data for more accurate classification of the data. However, more layers also implies higher latency and more storage than both the perceptron algorithm and SVMs.

K Nearest Neighbors (KNN) is a model that takes a new data point and gives it the same classification as the majority class of the K number of nearest training points. This requires no training, just remembering the training data and their classifications. Assuming that our training data can generalize to new data, this algorithm should perform quite well. The downside to this algorithm is that the latency is high since it takes a lot of computation to figure out the K nearest data points and also requires a lot of space to store all the training data. If we were to move away from using a computer to do the classification computations, the amount of memory required to store this information would need to be considered.

Random Forest is an algorithm that utilizes multiple decision trees that all operate on a subset of the data inputted. Each of these many decision trees will output a class for the data point and the class that appears the most among these decision trees is outputted as the final class. Because the decision trees within this random forest are uncorrelated with each other and use subsets of the data, the algorithm is more resistant to outliers and are less likely to overfit to training data. The disadvantage to this method, similar to KNN, is that it requires more memory to store each of the individual decision trees as well as needs more computational power to run all of the decision trees.

We have not decided which algorithm to use in our final product, but through preliminary tests, we have ruled out the perceptron algorithm. More discussion on the results of our preliminary testing in the Testing and Validation section (Section 6).

5 SYSTEM DESCRIPTION

5.1 Subsystem A - Glove

The glove subsystem consists of five flex sensors, an IMU, Arduino Nano and PCB.

5.1.1 Flex Sensors

When deciding what sensors to use to collect information on the shape of the hand, we considered placing IMUs on each of the joints on each of the fingers. If we decided on this strategy, we would need small enough IMU modules in a way which would not constrain the movements of the glove wearer. Although IMU chips are very small, making connections to them would be incredibly difficult without mounting them on a PCB. If the chips were mounted on a PCB, we felt that it would make the glove more bulky and less flexible. Thus we decided to use flex sensors. They would be easier to place on the glove and less bulky. Initially we had some concerns that the data from the flex sensors would not be enough to determine hand shape, but research into similar projects showed that flex sensors can be sufficient.

The two main flex sensors on the market are built by BendLabs and Spectra Symbol. We decided to choose the SpectraSymbol flex sensors over BendLab's for various reasons. The BendLabs flex sensor is very expensive at $50 per sensor for a 1-D sensor and $129 per sensor for a 2-D sensor. On the other hand, the SpectraSymbol flex sensors were only $13 per sensor. We have a budget of $600 and we needed at least five of these sensors, so we could not even purchase the 2-D BendLabs sensors. Additionally, the BendLabs flex sensors required six connections total to read data while the SpectraSymbol would only require three connections. Lastly, the BendLabs sensors had little documentation; there was only one tutorial about how to interface with the Sparkfun Pro mini. The SpectraSymbol sensors had been used in projects similar to ours and proved to be sufficient and had a lot more associated documentation on how to interface with different microcontrollers. To summarize, we decided on the SpectraSymbol sensors because (1) they were cheaper and we could buy multiple of them in case any of the five we needed got damaged (2) purchasing the minimum five sensors and three extras would still leave us a lot money left in our budget (3) they required fewer connections and we wanted to limit the bulkiness of our glove and (4) they had a lot more associated documentation.

5.1.2 IMU

There are a few ASL letters in which the finger poses are similar, but the orientation of the fingers make them different which is why we needed a way to determine the orientation of the hand. We decided to use an IMU to determine the orientation of the hand.

When researching IMUs to use, we found we could choose between purchasing a 6-DoF or 9-Dof IMU. A 6-DoF IMU has an accelerometer and gyroscope while a 9-DoF IMU has an accelerometer, gyroscope and magnetometer. The 6-DoF IMUs were cheaper and there was even one that we found which had some built in gesture recognition. However, we chose to go with the 9-DoF IMU since we found one that was marginally more expensive. Our research also revealed that 9-DoFs were more accurate since the addition of the magnetometer offset drift in the gyroscope. Furthermore, since we're feeding all of this data into a machine learning model, more data (the extra three data points from the magnetometer in a 9-DoF IMU) would be more beneficial. We chose the cheapest 9-DoF IMU we could find which was the Adafruit TDK InvenSense ICM-20948 9-DoF IMU. We had the option to just purchase just an IMU chip, however, we felt we did not have the skills or time to learn the tools to place it on a PCB. The product that we purchased included a breakout board with a 1.8V voltage regulator as well as level shifting circuitry to allow interface with 5V microcontrollers such as Arduino and Raspberry Pi.

5.1.3 Microcontroller

After selecting our sensors, we needed to figure out how to collect, read and parse the data so that it could be fed

into a machine learning model to classify the shape of the hand. At first we were not sure if we wanted a microcontroller to perform ML, or if we wanted a device to act as a gate reading from the sensors and feeding in the data. We decided we would use our laptops to run the models and use a small microcontroller to format and forward the data because we found that the microcontrollers recommended for running ML models were quite bulky and would be difficult to comfortably install on the glove.

There were various requirements for the device that would act as our gate and forward the sensor readings to the computer. The device needed to be capable of I2C or SPI to communicate with the IMU. It needed to have at least five analog pins to read from the five flex sensors. It also needed to be small. The Arduino Nano and RPi Zero were two devices which fit all these requirements, however, the Arduino Nano is smaller with a size of 45x18mm while the RPi Zero is 66x30.5mm. Additionally, Arduino is something that all of our team members were familiar with, so we chose to use the Arduino Nano.

5.1.4 PCB Design

In order to keep the circuitry compact, we decided to design a PCB to make all the connections instead of making the connections with jumper wires. Our PCB design is shown in Figure 5.

The circuit for utilizing the flex sensors with the Arduino Nano is shown in Figure 3. A flex sensor acts as a variable resistor. Its resistance changes depending on how much it is bent. By putting it in a voltage divider circuit, the Arduino can then read the fluctuating voltage in between the resistor and flex sensor to detect how much the flex sensor is bending. Since the flex sensors need to be placed along the fingers of the glove, the PCB simply has pinouts for the flex sensors to connect to as shown in Figure 5.

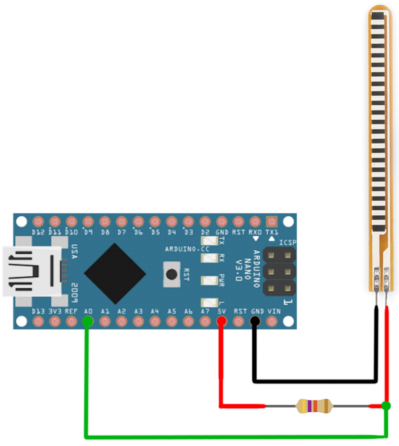


Figure 3: Flex sensor circuit and Arduino Nano connections

The circuit for connecting the IMU to the Arduino Nano for I2C communication is shown in Figure 4. Although the Adafruit TDK InvenSense ICM-20948 9-DoF IMU came with a QWiiC connection, we plan to communicate with it through I2C using its SCL and SDA ports in our PCB to reduce the amount of wires on the devices.

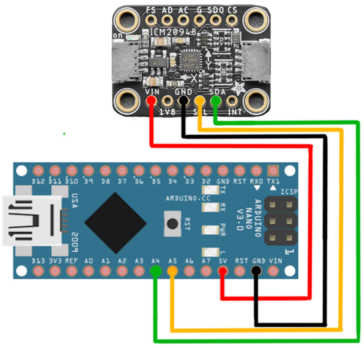


Figure 4: InvenSense ICM-20948 9-DoF IMU and Arduino Nano Connections

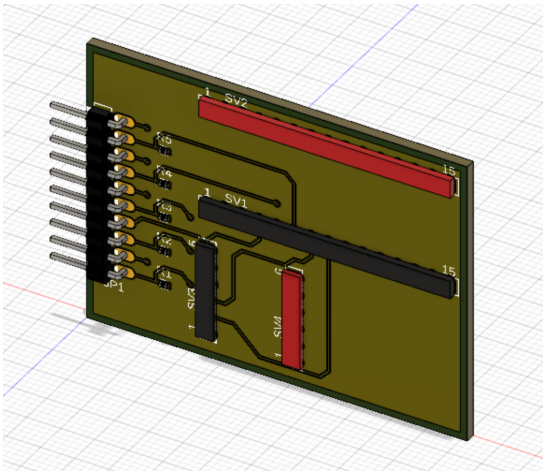


Figure 5: PCB 3D Model

5.2 Subsystem B - Computer

In terms of the Computer subsystem, we start off by reading the data that's passed to the Arduino Nano using Python. As the Arduino Nano will be connected to a computer using its serial port, we will use pySerial, a Python module for accessing serial port, to read data from the Arduino Nano. This module was chosen because it is

frequently updated library with ample documentation and testings.

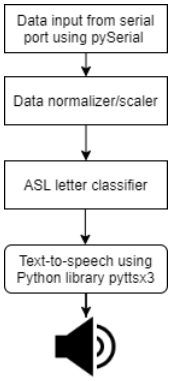
While the sensor data has been pre-processed on the Arduino to be more human readable, such as converting values from flex sensors to actual bending angles, further normalization and scaling will be done to ensure that the models can be trained correctly. This is a crucial step as certain models can have different results if data is not properly processed. This may also help in speeding up the calculations for model. The scaled data will then be passed through the classification model, which will output the most likely letter based on the given data.

As we are in the early phase of the project, we are still collecting data and have not determined the exact classification model to use. There are five models that we are currently considering: SVM, KNN, perceptron, random forest, and neural network. With our current set of data and fake data that we have generated prior to glove fabrication, we have found that random forest classifier gives the best result while the perceptron gives the worst. Other models' accuracy are in between.

We believe that perceptron is not doing well because it is simply linear regression and our data is likely not linearly separable. Random forest uses subset of data to train decision trees and takes the majority vote. The reason it works so well on our data may be because it uses subsets of data which can filter out the less relevant data in training and that decision trees tend to use the most important features to determine an outcome.

As for the other models that we considered, neural network, a widely used machine learning model, is not working as well as we expected. We deduct this is because our data does not contain many features since they are all 14 by 1 vectors. If we were to capture snapshots of data, neural network's performance may improve. As for KNN, k-nearest neighbor, this model finds the nearest neighbors of a data point and takes the majority value. This model has an accuracy of 82% on our first set of real data, which is already quite goood. However, as there are a few similar ASL letters and we are collecting more data in the future, this accuracy may decrease and its time on testing will increase, which will not be ideal if we want to meet our requirement for latency. SVM, support vector machine, is a linear model, but it is capable of performing non-linear classifications through data transformation. This is the model that performs the best after the random forest classifier. From our previous assumptions of nonlinear data, this may be the reason why it is doing well. Further analysis on models and data will be performed in the future to see if our assumptions hold.

After the classification model outputs the corresponding letter, we will then use Python's Text-to-Speech library, pyttsx3, to speak it out. Figure 3 (b) contains the above described system in a diagram.



(b)

Figure 6: System picture. (a) hardware subsystem (b) software subsystem

6 TEST & VALIDATION

6.1 Fake Data Generation

In order to make progress on our project while we waited on the parts for our glove to arrive, we decided to generate fake data to test the machine learning models we are considering. We began by first going through each of our gestures and determining a range of values for each of the 14 measurements our glove will take. For the flex sensors on each of the fingers, the data that is being sent to our machine learning models will be in the form of angles (in degrees). For the values outputted by the accelerometer and gyroscope components of the IMU, we were not able to find good documentation or sample data that could guide us in modelling our fake data, so we decided to take random values from the range that the product specification indicates will outputted. For the magnetometer component of the IMU, we chose the ranges assuming the user will always be facing one direction with respect to the Earth's poles. In reality will not be true, but it will at least provide our preliminary tests with some IMU data to work with.

After these ranges were determined, for each generated data point, we obtained a random value from a normal distribution within the range that we had set, with a small probability of generating an outlier value. Outlier values are simply random numbers from a uniform distribution between 0 and 100. We generated a total of 100 data points for each of the 26 letters for the training data set and 50 data points for each of the 26 letters for the testing data set.

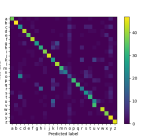
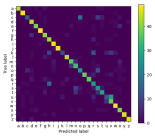
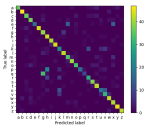
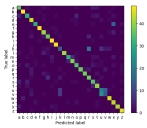
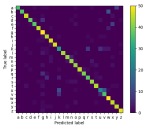
6.2 Preliminary Tests on Machine Learning Models

We did tests on the fake data we generated as well as real data collected from one person on the prototype of our glove after it was build. See Table 1 to see the classification accuracies of each ML model and Table 2 for the confusion

Table 1: Model Accuracy Comparison

Dataset	SVM	Perceptron	KNN	Random Forest	Neural Network
Generated Data	0.736	0.637	0.744	0.818	0.731
Real Data	0.873	0.760	0.821	0.968	0.798

Table 2: Confusion Matrices for each ML Model

Perceptron	SVM	Neural Network	KNN	Random Forest
				

matrices of each algorithm trained and tested on our generated data. All of these results should be viewed sceptically since our fake data generation left out important information for the data coming out of the IMU and there were some slightly loose connections in our prototype that likely affected data collection and will need to be fixed for the final product.

7 PROJECT MANAGEMENT

7.1 Schedule

Our schedule can be viewed in Figure 7. In general, we plan to spend the first half of the semester building the glove, and the second half of the semester training, testing and refining the ML model.

7.2 Team Member Responsibilities

Sophia is leading the hardware and construction part of the project. She is responsible for building the physical glove, making repairs and researching how to improve the design to create a robust, yet comfortable product. Her secondary responsibility includes gathering subjects for training/testing the ML models and helping conduct and compare experiments of competing design choices.

Rachel is primarily responsible for data collection, serial streaming of data, and normalizing the data as to reduce the noise from the data read in from the sensors. Her secondary responsibility also includes gathering subjects to collect training and testing data as well as analyzing the performances and costs/benefits of each ML model we are considering.

Stephanie is in charge of training and tuning the machine learning models. She also has a secondary responsibility of gathering subjects to collect training/testing data and helping determine how much data we want to read in and at what rate to get the best results (in terms of accuracy) without compromising the latency of our product.

7.3 Budget

The budget for our project is $600. So far we have only used $167.48 of it. A breakdown of our components purchased can be found in Table 3.

7.4 Risk Management

The primary risks to the success of our project involved the accuracy of our sensors. The sensors are how we detect the motion and pose of the hand. If those are not accurate, then the rest of the pipeline will not be accurate.

To mitigate the risk of faulty sensors, we bought multiples of the flex sensors, IMU and Arduino Nano. If any of these components get damaged during construction or testing, they can easily be replaced.

Further, after looking at the flex sensor specification sheets more in depth, the manufacturers recommend reading the voltage after passing it through an op-amp which acts as an impedance buffer. If our readings prove too unstable, this is a path we are prepared to take.

Lastly, we are normalizing the data the computer receives.

Another risk we anticipate is the PCB taking too long to get manufactured and delivered. In order to start testing as soon as possible, we have built the circuit out on perf board of about the same size as the design PCB. Once the PCB arrives, the glove circuitry can become cleaner with fewer wires.

8 ETHICAL ISSUES

Sign language interpreting gloves have been built in the past without input from ASL signers. It's important that ASL gloves accurately represent the language. For example, facial expressions are an important part of ASL, however that aspect of the language is inherently disregarded with a solution like a sign recognizing glove. We are well aware of this fault in our design and plan to consult actual ASL users during our development to collect and if time permits, integrate their feedback.

Table 3: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Microcontroller	Arduino Nano	Arduino	1	$14.98	$14.98
Flex Sensors	182	Adafruit	8	$12.95	$103.6
Pack of 12 Gloves	n/a	Donfri	1	$11.99	$11.99
A-Male to Mini-B USB Cable	n/a	AmazonBasics	1	$7.01	$7.01
9 DoF IMU	ICM20948	Adafruit	2	$14.95	$29.9
					$167.48

Note: Shipping costs are not included in the calculations.

Another ethical issue we faced when first figuring out our implementation details was privacy. As aforementioned, computer vision is a viable option for detecting and identifying ASL gestures. However, using a camera as part of our system could have privacy implications since anything that is captured in the scene could be a privacy violation. Since we decided to take a sensor approach instead, this issue is no longer relevant.

9 RELATED WORK

Gesture-recognizing gloves are not uncommon nowadays. During our research into implementation details, we have found quite a few similar projects. Here are a few that we feel closely resonated with our project.

Sign Language Glove[4] This project was built by two students from Cornell University. Similar to our objective, they wanted the glove to recognize ASL letters. Their finished product uses sensors to collect data, mainly flex sensors, an IMU, and contact sensors. They set up an off-glove circuitry on a breadboard to communicate between the sensors and the computer, which will train and test the data. The classifier models were then trained on data specific to each user.

The flaws within this design are quite obvious. The first and foremost being the glove is trained for specific users, and new users will have to train a set of data on it before they can use it. This greatly degrades user experience and can be confusing to new users if this were a commercial product, as users would likely expect a pre-configured product. The second drawback is the off-glove circuitry which adds complexity to the physical component and is inconvenient to set up and carry around. This also may impede movement while gesturing.

We are attempting to improve upon this project by finding a larger set of data to train on to generalize to a broad audience. We are also constructing a printed circuit board to aggregate our hardware components to help reduce the dimensions of our finished glove.

Sign Language Teaching Glove This project was done by students from University of Illinois Urbana-Champaign. This glove uses sensor-based detection, notably flex sensors for each finger, and gyroscopes and accelerometers at the end of each finger for angular motion and tilt sensing. Bluetooth transmitter was used to communicate sensor data to computer, which reduces the wires needed. Noise filter is applied to the data, which are then passed thru a perceptron algorithm for classification. Since extra sensors add more dimensionality to data that may increase accuracy, we considered this alternative. However, we found that the gyroscopes and accelerometers on the market are rather large and may be uncomfortable for the users and rejected this idea. We find it interesting that they chose to use the perceptron classifier despite having a rather low accuracy of 75%. We have decided to explore several different machine learning algorithms to find a model that's best fit for our type of data.

10 SUMMARY

So far, it seems we are on track to meet our design specifications. Our glove is lighter than 200g and our preliminary accuracy tests show we are about 96% accurate. However, we are still unsure how our glove will perform with different hand sizes and we will need to test the latency of letter recognition.

10.1 Future Work

We may continue our current work beyond this semester as this has been an interesting project so far. One area we can certainly improve on is the communications between the Arduino Nano and the computer. As of now, the glove uses a wired component for communications, which may not be ideal for users. We intend to change it into a wireless Bluetooth connection so that the users do not have to connect the glove to the computer to operate it.

Another area is certainly expanding the number of signs to recognize. This may prove to be a hard task as the physical components may need to be redesigned to include more sensors to detect a wider range of signs and to account for more complexity in signs.

10.2 Lessons Learned

The general advice that we have is to start on tasks early and follow through the schedule. If there is any slack in the schedule, we suggest still try to work on tasks that are scheduled for the future. If possible, we highly suggest dividing tasks between team members based on their expertise.

Another general advice to start early on contacting people for data collection. While we have planned enough time for data collection, we realized that it is something we could have started before the glove is built and doing it early could have saved us some time in the long run.

As for the technical side, our advice is to decide how the streams of data will be read in and if the user needs to signify pauses between signs using buttons or a pause sign. User experience and level of efforts needed will differ based on approach taken and that is something that needs to be balanced out in a time-constrained project.

We also suggest researching about different types of sensors. We had different type of sensors to use and different ideas of placement in mind. We believe that our current design will give the best accuracy for ASL letters while not sacrificing craftsmanship. However, there could be more optimal placements for other types of sign recognition and that is something worth of spending time to research about for future teams.

Glossary of Acronyms

- ASL – American Sign Language
- IMU – Inertial Measurement Unit
- ML - Machine Learning
- KNN - K Nearest Neighbors
- SVM - Support Vector Machine

References

[1] "Average Typing Speed Infographic — Ratatype." *Ratatype — Online Typing Tutor and Typing Lessons*, https://www.ratatype.com/learn/average-typing-speed/. Accessed 13 Oct. 2021.

[2] Deber, Jonathan, et al. "How Much Faster Is Fast Enough?: User Perception of Latency & Latency Improvements in Direct and Indirect Touch." *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 1827–1836.

[3] Bellugi, Ursula Fischer, Susan. (1972). A comparison of sign language and spoken language. Cognition. 1. 173-200. 10.1016/0010-0277(72)90018-2.

[4] Villalba, Monica Lin, Roberto. "ECE 4760 Sign Language Glove." *Electrical and Computer Engineering*, https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mjl256/webpage/. Accessed 13 Oct. 2021.

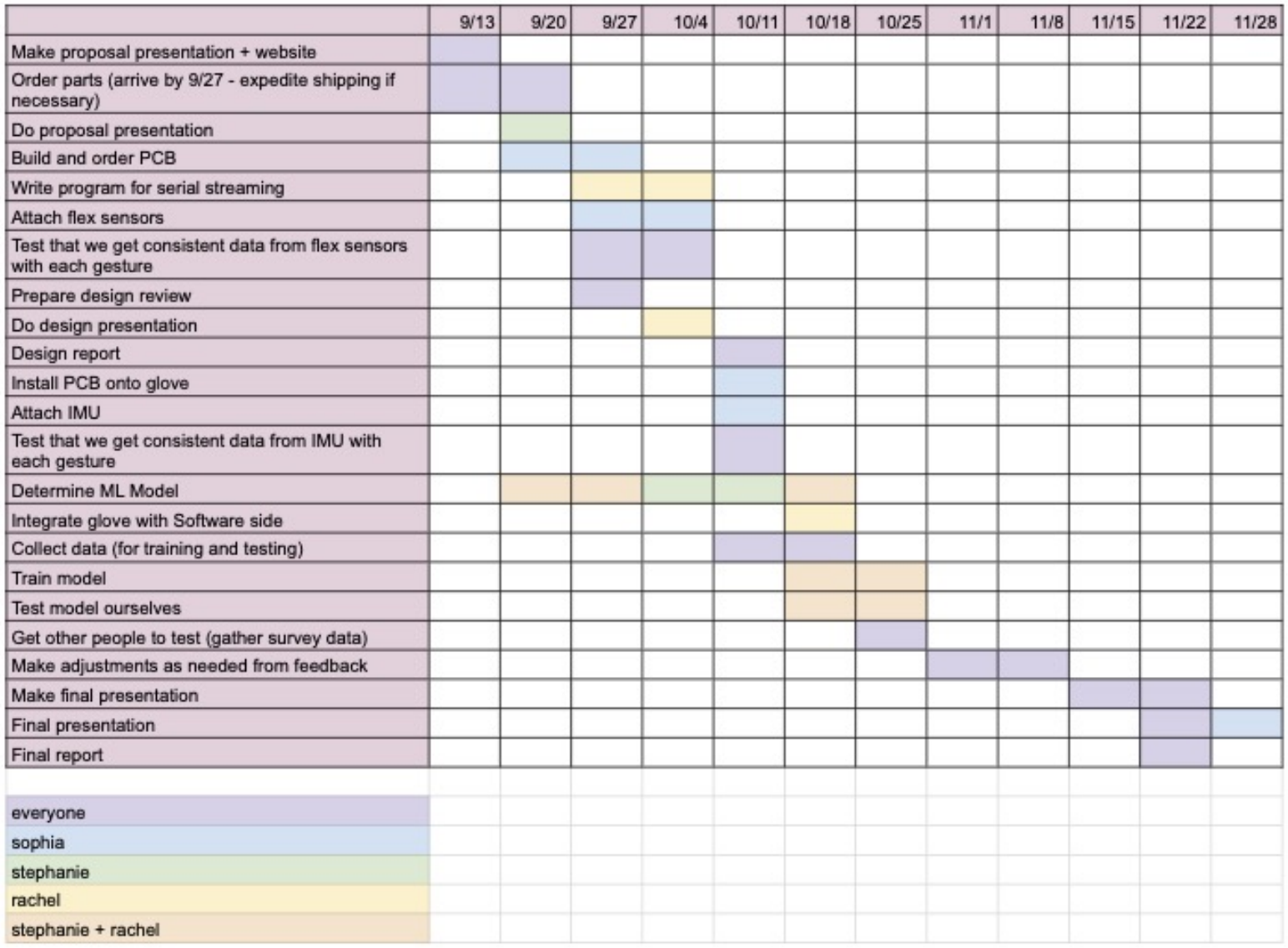


Figure 7: Gantt Chart