SENIOR CAPSTONE DESIGN DOCUMENT

OCTOBER 17,2019

# GESTURE RECOGNITION KEYBOARD

OREGON STATE UNIVERSITY
CS 461, FALL 2019

PREPARED FOR

SCOTT FAIRBANKS

|  |  |  |
|---|---|---|
| SCOTT FAIRBANKS | Signature | Date |

PREPARED BY

GROUP 11

|  |  |  |
|---|---|---|
| JEREMIAH KRAMER | Signature | Date |
| ZACHARY HORINE | Signature | Date |
| LAUREN SUNAMOTO | Signature | Date |
| CHANGKUAN LI | Signature | Date |

**Abstract**

This document outlines a detailed design that acts as a roadmap for the rest of our group's project. Specifically, this document describes how our solution will be implemented. Our project is broken up into manageable pieces where each group member is responsible for three pieces. Within the document, each piece is described in order to show relevance to the project. Further, the design of each piece is outlined.

## CONTENTS

## I. Introduction

### A. Purpose

This document will outline the development design and considerations of our project. Each of the following sections discuss the specific technologies and methods that we will use.

### B. Scope

This document will cover the overall design of our project, what systems and technologies we will use, as well as some of the research topics and theories that will help us as we build our project. This document is a plan and a snapshot of the project at its beginnings. The final project may use different tools, and may not employ the use of the same theories. This document has the ability to change.

### C. Overview

This document outlines the various pieces of our project. For each section, design decisions are made to identify the most viable option for the project, in terms of overall quality, stability, and ease of development. The topics covered include: platform, language, data storage, categorization and prediction engine, sensors, human computer interaction and character mapping. This document also covers research topics that will help with development, including: auto-complete, gesture studies and user memorization, ground truth finding, letter statistics, grammar statistics, and computer accessibility.

## II. Mobile Platforms

For our mobile application, there were two options we were considering for its Platform: iOS and Android. These are currently the two most prominent mobile platforms that exist. By examining each platform and considering key difference between them we determined that iOS was better suited to our requirements.

### A. iOS

iOS is a closed platform, with open source components and so, customization is limited. And yet, it is known to be very secure with the prioritization of privacy, security, and reduced risk of malware [1]. The platform's reputation for better security contributes to iOS having more penetration in the enterprise market [2]. All applications are purchased from the Apple App Store and software updates are available for older iOS devices. iOS has a higher revenue per user. iOS applications take less time to develop as code is written using Swift, Apple's official programming language. However, the process of publishing an application is time consuming because of strict controls put in place by Apple. Also, development and testing must be conducted on an iPhone. Lastly, this platform was requested by our client to be used [1].

### B. Selection

We will use the iOS platform mainly because it has been requested by our client to be used. Later, we may decide to launch the application on Android once it is established.

## III. Programming Language

The first piece of this project that I will address is the language of the software. The language that our team uses depends on the platform that we choose. Since we chose iOS as the platform of choice, I will discuss iOS-compatible programming languages. .

### A. *iOS Research*

Our options are: Swift, Objective-C, Javascript with React Native, Dart with Flutter, or C# with Xamarin [3]. Each of these languages has pros and cons. Swift is the most popular language used in most modern applications on the app store today. Further, it was developed and is maintained by Apple. Swift is also extremely robust and is always improving. Many libraries are available with Swift because it is an open source project open to anyone. Objective-C is the first language released by Apple back in 1984. This language is similar to C/C++ and many older apps are written in this language. Objective-C is well tested and is a stable language. A drawback of Objective-C is that it is losing popularity and there aren't as many Objective-C proficient developers. Javascript with React Native is a language and library combination. Javascript is a language mainly used for full stack development and web development. React is a library heavily influenced by Facebook. There are some drawbacks to this language. First, memory management is handled in a different way and there isn't a browser that will automatically handle memory efficiently. Second, performance is lacking compared to other languages. The main goal of React was to utilize a unified code base for both iOS and Android apps. However, the language has shown to cause more problems than it solves. React is in its early stages and will need to continue to get better. Dart with Flutter is a language and library combination created by Google. Dart is a language that is very similar to Java and C++ (object oriented languages). Flutter is a library that makes is very easy to create appealing-looking apps in a short amount of time. Google claims that this language is productive, approachable, and fast. This language is a newer language that is growing rapidly. Finally, Xamarin was created eight years ago. It is open source and is able to utilize ”hardware accelerations” that yields efficient performance. The drawback is that this language has decreased in popularity.

### B. *Recommendations*

Following programming language research for iOS, there are some good options to choose. I would suggest that we use Swift as our programming language. The reason for this is because Swift is a very popular language among iOS applications. This means that there is a lot of support from Apple and online communities.

## IV. Data Storage Method

In order to persist gesture data inside our app, we will need to store a set of gesture point arrays that encode the base data for each gesture, and allow the system to match the gesture that the user has input, and reference it to a character. There are many ways to do this, but the most commonly used are (a) an SQLite database, (b) external files or (c) as a set of key pairs stored inside the app configuration files. Both Android and iOS support all three of these methods, but they all have their specific use.

### A. *External Files*

External files are extremely useful when it comes to application design. sometimes, when the data that you have doesn't fit into a specific data model, it can be easier to store it in a format of your choosing and read it in yourself.

There are some drawbacks to this method though. Depending on where you store your data, the data can be accessed by the user and be moved or deleted. Although on all systems, there are private data stores that can be used to keep your data safe. These files often sit close to the application code so that they share the same permissions. One of the other drawbacks is that it is more difficult to keep track of your data, and where it is stored. This could be remedied by using one of the other data storage methods though, as it would provide a 'link' to the location of the data.

Overall, the concept of storing our gesture models in their own files makes the process of reading just one gesture in more efficient, and allows us to more easily add and remove data throughout the development process.

### *B. Key Pairs*

Key pairs are an easy way to store small amounts of data, and are useful for things like usernames or device keys. They are very specific on the types of data that they can store, and generally allow for personalized data to be stored without resorting to creating a whole database or file system.

On iOS, these keys are stored in a '.plist' file within the app package. Because of the way the file is stored, there is a maximum file size of 4GB (file-system restriction). [4] This, along with the fact that the file has to be read in every time the app loads, or the first time data is requested, means that the amount of data needs to be limited, and shouldn't be used to store large data structures. The only difference with Android devices is that the 'Shared Preferences' objects are stored in an xml file within the system. [5] Otherwise they act the same as an iOS device.

### *C. Selection*

For our application, the best way to store our data actually comes as a combination of two methods. We will be using both external files, as well as key pairs stored inside the application executable. The advantage to this is that we can store our gesture models in their own, individual files, which allows us to make them modular, and enables the easy creation of user gesture files. We will then use key pairs to store the association between the character and their gesture file. This also allows us to avoid some of the downfalls of external files.

## V. Categorization engine - how to find a ground truth

### *A. What is a Ground Truth?*

In general, a ground truth is a data set that defines the base observations that all observations made after its establishment are based on. The concept of a ground truth is used extensively in image recognition, and is heavily related to a scientific assumption. In this way, it can be used as a basis of observation, and allows the observer to determine whether or not the data that they have gathered fits the model or not. [6]

### *B. What is included in a ground truth?*

At its base, the ground truth is the simplest, most basic, and neutral form of the data expected. However, one set of perfect data is not enough to define the base case of the model. A ground truth has to not only include the optimal solution, but it also has 'sub-optimal' solutions that are still in the threshold. [7] It also includes solutions that are not in the data-set, which help to filter out data that doesn't fall within the confines of the desired outcome.

In image recognition, the ground truth also has multiple layers. All of the layers help to filter the data, and make it easier to pinpoint what is different from the ground truth. By looking at things such as the diffusion, reflectivity, shading and the specular components of an image, it is easier to see what part of the image differs from the ground truth, which makes it easier to correct later. [8]

### C. How can we apply this?

Detecting gestures is difficult for many reasons, but they all lead to one thing: no two movements are ever the same. A lot of this has to do with sensor accuracy, as the accelerometer in a phone is 'precise' but not very accurate. The little irregularities in motion sensing stack up, and very quickly make pulling the true motion out of the noise very difficult. The other complication is that humans never recreate the same exact gesture, and no two humans will perform the same gesture the same way. because of this, the motion itself is not accurate to the model. This is why a simple 'perfect' gesture can't be used as the ground truth. By using a ground truth in multiple axis, as well as tolerances and negatives, we can eliminate bad data, and generate the most probable path.

## VI. Prediction Engine

The goal of the prediction engine is to intelligently match specific user input to its corresponding character mapping. In order to implement this engine, we need to select and develop an algorithm.

### A. Algorithm Background Information

This algorithm will be a machine learning algorithm. This means that, the algorithm needs to predict a possible output based on user input. Of the types of machine learning algorithms, this algorithm that our team needs is a "supervised learning" algorithm as we need it to find a target or outcome variable predicted from independent variables generated by user input [9]. The independent variables will be inputted into this algorithm that generates desired outputs. This algorithm model will need to be trained to generated the desired outcome until we reach a high level of accuracy.

### B. Motion Gesture Algorithms

Since there are multiple different supervised learning algorithms that we can use, we need to find possible algorithms that relate to motion gestures the closest. In an academic journal named "Motion-Based Gesture Recognition Algorithms for Robot Manipulation," there are three algorithms that are discussed. These algorithms include: Dynamic Time Warping, Hidden Markov Model, and Distance Metric Learning [10]. The journal assesses the accuracy of these accelerometer-based algorithms. Dynamic time warping and hidden Markov model algorithms are two classical pattern recognition methods.

*1) Dynamic Time Warping:* The dynamic time warping algorithm is a geometric approach to predicting an output. Fundamentally, the algorithm uses two different time series and will "warp" one time series to make it resemble the other series. When warping, the goal is to make the accumulative distance between them minimal. The more the algorithm succeeds in its goal, the more accurate the algorithm is at predicting the desired outcome. With this algorithm, dynamic programming is used. The output is a matrix that indicates how much the two different time series differ when they are perfectly aligned. This algorithm fits perfectly with motion gesture application because it can use a set of known pattern templates and recognize an unknown pattern based on the known templates. It chooses the template that best fits one of the known patterns. Our team would leverage this by mapping specific gestures to a corresponding character.

*2) Hidden Markov Model:* The hidden Markov model algorithm is a statistical approach to prediction. The hidden Markov model is an extension of the observable Markov model. An observable Markov model is a discrete, first order Markov chain that has: a set of states, a set of state transition probabilities, and a specification of an initial state. "A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules" [11]. The output of the model's process is a set of states at an instant of time and each state corresponds to a certain physical event that is observed [10]. The process becomes ”hidden” when the observation is a probability of the state. Therefore, the state isn't directly observed. In a nutshell, based on specific probabilities of states at given times, a physical event can be predicted. This fits well with motion gesture recognition. Since the hidden Markov model can have different typologies and models, the model that fits the best with motion gesture recognition is a model that describes observations in a chronological sequence. This model is named the left-to-right model. With this algorithm, our team would save specific patterns that correspond to a character mapping by setting precise probabilities of space and time.

*3) Distance Metric Learning:* The distance metric learning algorithm is a newer technique that recognizes patterns based on a group of known patterns. The unknown pattern is then assigned to the closest known pattern. The difference in the distance metric learning algorithm is the underlying closeness metric distance from Euclidean distance to the Mahalanobis distance metric. This new metric provides more accurate results. The drawback to this algorithm is that it is very complex compared to the other algorithms.

### *C. Review and Recommendation*

The academic journal that suggested these three algorithms performed tests on each of them separately and reviewed their accuracy. The distance metric learning performed with 98.26% overall average recognition rate. The hidden Markov model achieved 94.44% overall average recognition rate and the dynamic time warping algorithm achieved 89.89% overall average recognition rate. In order to decide the algorithm to choose and implement, my team needs to weigh algorithm performance versus its complexity. This means that our team may choose an algorithm that doesn't yield the highest accuracy if another algorithm is easier to implement. The reason for this is because of overall time constraints for our project. With that being said, our team should wisely start with the distance metric learning algorithm or hidden Markov model.

## VII. Sensors and interpretation of output data

In order for the gesture-based keyboard to work certain sensor must be utilized and inputs must be recognized reliably.

### *A. Option*

This section outlines two sensors as well as two methods to record data from them. The first sensor is the gyroscope, the second sensor is the accelerometer. The two methods of recording data from these sensors are calibrated and non-calibrated.

### *B. Goals for use in design*

The main goals for the use of these sensors and data recording methods is accessibility and accuracy. These methods should be able to be used on every smartphone as well as be accurate to allow for complex gestures. The speed at which the user can type should be based off their skill rather than hardware or software limitations.

### *C. Criteria being evaluated*

The sensor must be abundant in modern smartphones and the data recording method should allow for accurate and smooth reading.

### D. Comparison Breakdown

- Accuracy, the whole project is based around gestures that need to be reliably recognized by the smartphone.
- Accessibility, the method of data recording and processing should not depend on a piece of hardware that is difficult to come by.
- Speed, hardware and software limitations should not hinder the user experience.

1) Gyroscope
    - Gyroscopes accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
    - Gyroscopes are present in near all modern smartphones.
    - Gyroscopes output raw data instantly.
2) Accelerometer
    - Accelerometers accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
    - Accelerometers are present in all modern smartphones.
    - Accelerometers output raw data instantly.
3) Uncalibrated
    - Uncalibrated data is raw and is precise but not accurate.
    - Uncalibrated data is outputted the same on all smartphone gyroscopes [12].
    - Uncalibrated data is recorded instantly.
4) Calibrated
    - Calibrated data accuracy is dependent on the fidelity of the calibration.
    - Calibrated data is dependent on factory calibration methods [12].
    - Calibrated data is recorded instantly.

### E. Selection

The best option is actually to use the Gyroscope and Accelerometer in tandem and to record data uncalibrated data and manipulate it. Calibrated data recording relies heavily on the accuracy of the factory calibration methods and results in most jumpy data. This is due to the fact that it tries to account for the faults in the sensor and guess at the correct output [12]. Uncalibrated data recording allows for less sensor noise [13] which means that the data is less likely to have outliers or jumps. This allows for a smoother reading if averages of the output data are taken over short intervals. By using both the gyroscope and accelerometer is it possible to differentiate between the device being upside down performing a gesture vs right side up doing the inverse gesture. Additionally, creating artificial zeros at the start of every recording session allows for an artificial calibration which in turn results in the best recording method for this particular application.

## VIII. Human Computer Interaction

The Human Computer Interaction is one of the most important aspects of the project due to its nature. The user is using a gesture-based keyboard and needs to understand what they are inputting at all times.

### A. *Option*

This section outlines three methods of Human Computer Interaction that could be used in the project. The first is haptic feedback, the second is sound queues, the third is on screen prompts.

### B. *Goals for use in design*

The goal of Human Computer Interaction is to provide the user with quick and easily discernible feedback on what is being inputted. In this application the user is manipulating their phone in order to input characters so it is important that the user has a way of receiving feedback regardless of the position of the phone.

### C. *Criteria being evaluated*

The most important criteria that these technologies are being critiqued by is their speed and accessibility. This method of keyboard input does not allow for the user to be looking at the smartphone screen at all times.

### D. *Comparison Breakdown*

- Speed: One of the requirements of the project is for a user to be able to type at an acceptable speed.
- Accessibility: The technology should have the least amount of hoops for the user to jump through in order to be used.
- Abundancy: The technology must be present in most all smartphones to allow for a broader user base.

1) Haptic Feedback
    - Haptic Feedback is instant upon user input.
    - Haptic Feedback works without the user having to access and settings or plug anything in.
    - Haptic Feedback is present in almost all smartphones as it uses the vibration motor.
2) On Screen Prompts
    - On Screen Prompts, are instant but users need time to read them.
    - On Screen Prompts, are dependent on the size of the screen.
    - On Screen Prompts, are the most accessible and are present on every smartphone.
3) Sound Queues
    - Sound Queues are instant upon user input.
    - Sound Queues require the user to be in an area that allows it and or the use of headphones.
    - Sound Queues are present on every smartphone.

### E. *Selection*

The best option for this project is Haptic Feedback. Haptic Feedback is perfect for this application as the users hands must be on the device to begin with which guarantees that the user will receive the vibration. This also allows for the phone to be in an orientation in which the user cannot see the screen which opens doors to more gestures. Haptic Feedback solely relies on the vibration motor of a smartphone and can be fired instantly to correspond with a specific gesture [14].

## IX. Human Computer Interaction: Computer Accessibility

Computer accessibility in human-computer interaction refers to the accessibility of a computer system to people despite disability type or severity of impairment. In developing our application we hope to improve the accessibility of text input by creating features that prioritize individuals with specific visual, hearing, and motor or dexterity impairments.

### *A. Visual Impairments*

Visual impairments include blindness, low vision, and color blindness [15]. In developing our application we can build features that rely on other senses, such as hearing and touch. We can use haptic technology which involves sending a user feedback by creating vibrations in a mobile phone. Such feedback is important in recall and recognition of users as they learn to use the application. For example, a vibration can alert a user to the generation of a letter and aid the speed in which they ”type”.

### *B. Hearing Disabilities*

Hearing disabilities range in severity from total deafness to slight loss of hearing [15]. As mentioned above, sound queues would not be very effective. Therefore, haptic technology would be a better alternative method to provide user feedback.

### *C. Motor or dexterity impairments*

Motor or dexterity impairments encompass a large variety impairments including total absence of limbs or digits, paralysis, lack of fine control, instability or pain in the use of fingers, hands, wrists, or arms [15]. Some of these individuals would especially benefit from alternatives to standard input methods (i.e. keyboard). Although, not all individuals that fall under such category will be able to use our application because it requires some fine motor skills for precise generation of characters as well as gross motor skills with wrist movement. But, there are specific elements of our application that we can do differently from traditional keyboards.

*1) Touchscreen Manipulation:* Traditional mobile keyboard require steady and precise tapping on small key spaces which can be difficult for certain users, such as people with Arthritis or those with uncontrollable hand tremors from Parkinson’s disease or Multiple sclerosis [16]. Therefore, our application should not require complex use of the screen. For example, in addition to defining gestures for alphabet letters we should define a gestures for deleting and shifting. It should also be error-tolerant (e.g. deletion should not necessarily be easy to do).

*2) Customization:* To accommodate the variation in impairments we should allow for some customization of our application. For example, the typing speed should be easily adjustable. The transition between characters/gestures can be adjusted as well as the recognition and translation of characters/gestures.

### *D. Conclusion*

In developing our application we will consider the importance of computer accessibility by creating features that meet the needs of certain individuals with specific disabilities. The functionality of our application will rely mostly on gesturing and user feedback will include haptic feedback which usable for a wide range of users.

## X. Gesture Studies and User Memorization

Gesture Studies and user Memorization outlines the user experience in terms of the speed of learning. This differs from the Human Computer Interaction Section as it is a system that is not necessary once the keyboard has been learned rather than one core to the project.

### A. Option

This section outlines the two different methods of feedback that will help users memorize the gesture-based keyboard and increase their typing speed.The first is sound queues and the second is on screen visuals.

### B. Goals for use in design

The goals for the gestures for the project is for them to be easily differentiated and simple to execute. This means that there has to be clear feedback to the user as to what they are inputting and the options that they have going forward.

### C. Criteria being evaluated

The most important aspect of Gesture studies and user memorization is the speed of learning and simplicity. This means that the speed, learning speed, and accessibility are all criteria that will be used to evaluate it.

### D. Comparison Breakdown

- Speed, the feedback to aid memorization must feel that it is tied with user inputs so it must be instant.
- Learning Speed, the system must be easy to understand and memorable.
- Accessibility, the system must be within the capabilities of the modern smartphone.

1) Sound Queues
    - Sound Queues are instant.
    - Sound Queues are among the most easily recognized feedback methods and are highly recognizable [17].
    - Sound Queues can be played by all modern smartphones.
2) Visual Aid
    - Visual Aids are instant but take time for the user to process.
    - Visual Aids can be easily learned.
    - Visual Aids can be displayed on all modern smartphones.

### E. Selection

The best option for this project is sound queues. Though the use of haptic feedback is ideal for the core elements of the project, there is still the aspect of auto-complete and predictions that have to be conveyed to the user in an efficient manner. This can be best achieved by using sound clues. RCP Tones offers free use of their sound effects for app developers but it is extremely important that the selection of the sounds is strenuous [18]. There is a whole science behind how the brain memorizes things but the field of sound is the most straightforward. The brain memorizes melodic patterns the easiest, after just 10 playbacks of a patterned sound the average person can pick out the sound for other similar ones [17]. By choosing distinct, short, pattern-like sounds users can easily understand what the program is trying to convey to them. The applications of this include letting the user know that there is an auto-complete ready for them, as well as telling the user what group of letters they are selecting. This paired with an auditory output upon the selection of a letter should create a flow in the user experience which allows for fast learning.

## XI. Auto-complete

### *A. Option*

This section outlines two options for auto-complete. The first being word based, the second be letter based.

### *B. Goals for use in design*

The goal for auto-complete is to increase user input speed without hindering user experience in anyway through perfect implementation.

### *C. Criteria being evaluated*

The most important aspects of the auto-complete method is speed and the lack of unwanted prompts. The user must feel like they have the option to accept the auto-complete without it getting in the way of typing the next letter.

### *D. Comparison Breakdown*

- Speed, the auto-complete must output a suggestion before the user can input the next letter.
- Accuracy, the auto-complete must be able to adapt to the user and not repeatedly suggest a word that the user does not use often.
- Learning Speed, the user must be able to predict what the auto-complete will suggest through typing experience.

1) Word Based Auto-complete
    - Word Based Auto-complete is marginally slower than letter based.
    - Word Based Auto-complete is accurate for common words.
    - Word Based Auto-complete is fairly unpredictable unless a longer word is being typed.
2) Letter Based Auto-complete
    - Letter Based Auto-complete is near instant.
    - Letter Based Auto-complete is based off the probability of what the next letter will be [19].
    - Letter Based Auto-complete is unchanging and predictable with practice.

### *E. Selection*

The best option for this project is Letter Based Auto-complete due to its simplicity and ease of implementation. The gesture-based keyboard is outlined purely in letters and implementing a way to enter a word based off auto-complete could raise problems. Additionally, being as letter based auto-complete is derived from the likelihood of the next letter, it allows for the elimination of others if their probability becomes zero. By using deterministic acyclic finite state automaton (DAFSA) [19], a string of letters leads to the next which allows for the user to be guided to their next letter with more accuracy then the patterns in human speech that word based auto-complete uses. For example, DAFA outlines that if a Q is inputted by the user then the next most likely character would be U [19]. This allows for the user to also learn the auto-complete method and utilize it to master the input method.

## XII. Character Mapping

Character mapping is extremely important because it affects user experience and algorithm success. Since the goal of our application is to create alternative user input via motion gestures, our team needs to map motions to letters and words. Given this, we have some different options for implementing character mapping.

### *A. Straight Line Simple*

One approach is to map 30 unique gestures to 30 different characters. The reason we chose 30 is to allow for 26 different letters in the English alphabet and four additional gestures for special characters such as periods and commas. In order to make it easier for the user to learn the gestures, the gestures would be simple straight line movements of their device. However, mapping 30 straight line gestures would be an issue because 360 degrees divided by 30 is only 12 degrees of space for the user to attempt to generate text. This would be infuriating for the user because the degree for error is much too small. To take this a step further, our application could leverage the gyroscope embedded in mobile devices and initially flip the device on its side, facing upright, or facing downward. Then only 10 movements in space would be needed. That means that users would have 36 degrees (360 degrees divided by 10) of space on the devices axis to gesture. This is one possibility. However, if users wanted to add shortcuts as new gestures, this would limit them.

### *B. Series of Combinations*

Another approach that our team can take is to map a series of gestures to certain characters and words or phrases. In other words, if the device is moved forward, right, right in succession, that would result in a corresponding letter. With this idea, the user would only have to move their device in six different directions: forward, right, left, backward, up, and down. This would allow our algorithm to be more successful. However, the drawback is recognizing when the next ”gesture” occurs (either within the combination series, or between characters). In order words, from the previous example, the algorithm would need to know when the user stopped gesturing forward, and started gesturing to the right. Also, the algorithm needs to know when to select a character and allow for the user to gesture the next character. Further, the amount of combinations could become complex, especially if the user adds many shortcuts. This would make it difficult for the user.

### *C. Object Pattern Complex*

Another idea is to gesture specific objects and map them to their respective character. For example, gesturing a ”circle” would result in the letter ”o.” This idea came from the academic journal that was referenced prior [10]. Although this is a valid approach, I think this would be very complex for the user. Even though this might result in a higher accuracy for the algorithm, text generation speed would suffer greatly.

### *D. Review and Recommendation*

After review, mapping gestures to characters, words, or phrases, is difficult for users and the algorithm. As a team, we know that this will be a challenge and users and the algorithm will need to continuously learn from one another. With that being said, users should be able to map custom gestures to the application. This would result in a more user friendly application and increased usability. Our application should have default character-gesture mapping and allow for custom gestures. Also, the user should be able to edit the default mapping. The default mapping should be only moving the phone in simple straight lines with the phone’s initial position altering.

## XIII. Letter Statistics

This application must be able to generate text from motion gestures with a mobile device. We will define a set of gestures that is easy to complete and replicate, and allows for a 'typing speed' of at least ten words per minute. We will include gestures for the thirty-size standard characters, as well as a include support for various modes of punctuation, white-space and capitalization. Letter statistics can be useful in determining how we define the different gestures. Furthermore, like the optimization of keyboard layout letter statistics can be used to improve typing speed and decrease user errors.

### *A. Letter Frequency*

To ensure our goal 'typing speed' we should prioritize the most commonly used alphabetic characters, and pair the least used characters with gestures that are more complex due to the need for distinct motion-based gestures. Based on a sample of 40,000 words, the letter 'Z' which occurred 128 times and is the least frequently used English letter should be defined in a way such that it is not easier to replicate than the most frequently used letter, 'E', which occurred 21,912 times. Also, using these letter statistics we can consider how the letters relate to one another. It is easier to identify vowels because most letters appear before and/or after them [20]. Furthermore, our application should be error-tolerant by defining the backspace character so that it cannot be misinterpreted as a common character gesture.

### *B. Digraph Frequency*

In addition to letter frequency, we can look at the statistics for digraph frequency to improve user 'typing speed' as well as accuracy and ease of use. Digraphs are pairs of letters. For example, based on a sample of 40,000 words, the most common digraph is ”th” which occurred 5,532 times. This is consistent with the fact that the word, ”the” is the most commonly used word in the English language. This information is useful in defining motion-based gestures because it is important to consider the transition between letters, especially those that are frequently next to each other. We must consider that there are specific gestures that will be difficult to create and discern following another gesture [21].

### *C. Conclusion*

By using letter and digraph frequency statistics in the defining of our motion-based gestures, we will be able to maximise the efficiency of typing on our keyboard. The most common alphabetic characters and digraphs will be prioritized over the least common. Overall, such considerations will help in improving user experience by prioritizing ease of use and design.

## XIV. Grammar statistics (frequencies and common patterns)

### *A. Keyboards*

One of the important things when designing an input device is the layout of the input 'keys'. Many traditional keyboard layouts were designed around the English language, and the letter frequencies that are used to type some of the most common words. One of the most common examples of this is the Devorak keyboard, which prioritizes vowels and commonly used consonants. [22] we will have to do a similar thing with our keyboard, but we will also have other issues to attend to. Because of the nature of gestures, there are certain gestures that will be difficult to detect if they come after another gesture, and some that are easy and flow into one another. In this way, we are dealing with some of the issues that came along with typewriters (frequent jamming), and the supposed reason that the qwerty layout was invented. In reality, the layout was invented to slow down faster typists, which is exactly the opposite effect that we want to have. [23]

### *B. Frequency in Words*

Instead, we should pair more readily accessible gestures with the most commonly used characters, and associate the least used characters with ’more out of the way’ gestures. We also need to look at the characters that commonly come after one another, and make sure the par of gestures work well together. A study done by the College of Math at the University of Illinois shows how likely a given character is to come after another. In their study, there were a large number of characters that never appeared together, and a list of the most frequent ones, which included ’th’, ’he’, and ’an’ as the top three. [24] This makes a lot of sense, as the three most common words in the English language are ’the’, ’of’ and ’and’. [25]

### *C. Conclusions*

If we can use this information to map our gestures to letters, then we can improve the efficiency of typing on our keyboard. By taking into account all of the lessons learned in keyboard design, and incorporating them into our app, we can create a better experience for our users.

## XV. Conclusion

This document covered the overall design and considerations of our project including the selected technologies and methods that are best suited to our project’s specific requirements.

## REFERENCES

[1] S. Aggareal, “Android vs ios: Which mobile platform is best for app development?” techahead. [Online]. Available: https://www.techaheadcorp.com/blog/android-vs-ios/

[2] T. Manifest, “Android vs ios: Which platform to build your app for first?” medium. [Online]. Available: https://medium.com/@the_manifest/android-vs-ios-which-platform-to-build-your-app-for-first-22ea8996abe1

[3] E. Chung, “What language are iOS apps written in?” [Online]. Available: https://www.zerotoappstore.com/what-language-are-ios-apps-written-in.html

[4] “iOS Data Storage Guidelines,” 2019. [Online]. Available: https://developer.apple.com/icloud/documentation/data-storage/index.html

[5] Tarun Kumar, “Top 10 ways you should know to store data in Android,” Mar. 2018. [Online]. Available: https://www.loginworks.com/blogs/top-10-ways-know-store-data-android/

[6] Danilo Pena, “Ground Truth Versus Bias,” Apr. 2018. [Online]. Available: https://towardsdatascience.com/ground-truth-versus-bias-99f68e5e16b

[7] S. Krig, *Computer vision metrics: survey, taxonomy, and analysis*, ser. The expert’s voice in computer vision. New York, NY: Apress, 2014, oCLC: 881828978. [Online]. Available: https://www.embedded-vision.com/sites/default/files/apress/computervisionmetrics/chapter7/9781430259299_Ch07.pdf

[8] Roger Grose, Micah K. Johnson, Edward H. Adelson, and Willian T. Freeman, “Ground truth dataset and baseline evaluations for intrinsic image algorithms,” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Feb. 2009. [Online]. Available: https://www.cs.toronto.edu/~rgrosse/iccv09-intrinsic.pdf

[9] S. Ray, “Essentials of Machine Learning Algorithms (with Python and R Codes),” 09-Sep-2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

[10] Tea Marasović, Vladan Papić, and Jadranka Marasović, “Motion-Based Gesture Recognition Algorithms for Robot Manipulation,” vol. 12, no. 5, p. 51, 2015.

[11] Henry Maltby, “Markov Chains,” https://brilliant.org/wiki/markov-chains/, 8-November-2019.

[12] A. Developers. Motion sensors. Android. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion#java

[13] B. by Xsens. Rms noise of accelerometers and gyroscopes. Xsens. [Online]. Available: https://base.xsens.com/hc/en-us/articles/115000224125-RMS-noise-of-accelerometers-and-gyroscopes/

[14] L. Critchley. How do haptic sensors work? AZO Materials. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=15700

[15] “Disabilities affecting computer accessibility,” Forth ICS. [Online]. Available: https://www.ics.forth.gr/hci/ua-games/disabilities.html

[16] “Motor disabilities: Types of motor disabilities,” webaim. [Online]. Available: https://webaim.org/articles/motor/motordisabilities

[17] Z. Macintosh. Brain quickly remembers complex sounds. LiveScience. [Online]. Available: https://www.livescience.com/10670-brain-quickly-remembers-complex-sounds.html

[18] R. Tones. Dev_tones ui and menu sounds for your app. RCP Tones. [Online]. Available: https://rcptones.com/dev_tones/

[19] J. Daciuk. Comparison of construction algorithms for minimal, acyclic, deterministic, finite-state automata from sets of strings. Implementation and Application of Automata Lecture Notes in Computer Science. [Online]. Available: http://web.cs.mun.ca/~harold/Courses/Old/Ling6800.W06/Diary/q3p2qx4lv71m5vew.pdf

[20] “English letter frequency (based on a sample of 40,000 words),” Cornell. [Online]. Available: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

[21] “Digraph frequency (based on a sample of 40,000 words),” Cornell. [Online]. Available: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/digraphs.html

[22] Randy Cassingham, “The Dvorak Keyboard,” 2019. [Online]. Available: https://www.dvorak-keyboard.com/

[23] Leah Welborn, “Why QWERTY?” Jun. 2011. [Online]. Available: https://www.mentalfloss.com/article/27938/why-qwerty

[24] Jeffrey S. Leon, “Frequency of Character Pairs in English Language Text,” University of Illinois, Tech. Rep., 2008. [Online]. Available: http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/char_freq2.pdf

[25] “100 Most Common English Words (Learn 85% of English in 100 Days),” 2019. [Online]. Available: https://www.rypeapp.com/most-common-english-words/