Development of a finger-key identification submodule for a touch typing trainer

---

A Special Project Proposal Presented to the
Faculty of the Department of Computer Science,
University of the Philippines Cebu

In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Computer Science

Oscar Vian L. Valles
BS Computer Science

Dhong Fhel K. Gom-os
Adviser

February 2022

Acronyms

ANSI American National Standards Institute

CTS Carpal Tunnel Syndrome

FP ACC Finger Placement Accuracy

HFP ACC Historical Finger Placement Accuracy

ISO International Organization for Standardization

JIS Japanese International Standards

ROI Region of Interest

WPM Words per Minute

WRNULD Work-related neck and upper limb disorders

List of Tables

List of Figures

Contents

Chapter 1

Introduction

1.1 Background of the Study

There are a lot of educational typing tests available that help people learn touch typing, including Monkeytype, TypeRacer, and Keybr. These typing tests list out words that are then typed out. The entered keys are then compared to check if the user has typed the expected letter. At the end of the test, the time taken is calculated, and certain metrics are given. These metrics include Words per Minute (WPM) and Accuracy (Bartnik, 2021).

However, this method of examination leaves out a crucial part of typing — ergonomics. Ergonomic typing prevents a lot of health issues in the future like repetitive strain injury or carpal tunnel. One important factor that affects ergonomics is the typing procedure and posture. This means the proper placement of the wrist, hands, and hitting the keys using the right finger that is assigned to the key.

Correct finger placement is usually taught at the beginning using a diagram, with each key being associated with a specific finger. For instance, the letter Q in a QWERTY layout should be hit using the fifth digit, or the little finger, of the left hand, and this is shown by coloring the fifth digit and the key Q with the same color or by placing the letters directly on the fingers (Dobson, 2009).

Incorrect finger placement may cause these hand and wrist positions: ulnar deviation, forearm pronation, and wrist extension (Serina et al., 1999). These three are hand and wrist positions that are common in all activities, however, prolonged periods in these positions may cause injuries such as Carpal Tunnel Syndrome (CTS) (Toosi et al., 2015)

In addition, this type of typing is frequently taught in the beginner level (Donica et al., 2018). This means that there is a need to weed out bad habits that may develop, like using the index finger for pressing the spacebar or backspace. However, it is impractical for an educator to check each student if they are not performing these movements as these may only show for a small period which may not be caught in time.

Thus, there is a need for automatically detecting which finger is used during typing. One way to do this is through finger and hand tracking. One solution for tracking is by using image processing and machine learning. An example of this is MediaPipe by Lugaresi et al., 2019.

MediaPipe allows for various applications for machine learning in the field of image processing. This includes, hand tracking, pose estimation, object detection, and others. Another example of a library that allows for hand and finger tracking is OpenCV by Bradski, 2000. This is a tool that simplifies computer vision and image processing. Machine learning can also be used with OpenCV.

1.2 Research Questions

1. What algorithm or technique using computer vision is capable of identifying which keys are being pressed by which fingers during keyboard typing?
2. What setup and configuration of a single optical camera can be used to capture images during keyboard typing for real-time finger-key identification using the algorithm identified in 1?
3. How to design and develop a touch typing trainer using the algorithm in 1 and camera setup and configuration in 2?
4. How accurate is the touch typing trainer in identifying which finger is used to press which key?
5. What new keyboard typing metric can be developed in order to consider both the key and finger used to press as parameters?

1.3 Research Objectives

1.3.1 General Objectives

To create a touch typing trainer that detects poor finger placement and hand position to help develop better typing habits and healthier typing ergonomics.

1.3.2 Specific Objectives

1. Develop an algorithm or technique using computer vision that is capable of identifying which keys are being pressed by which fingers during keyboard typing — i.e. finger-key identification.
2. Identify the setup and configuration of a single optical camera that can be used to capture images during keyboard typing for real-time finger-key identification using the algorithm identified in 1
3. Design and develop a touch typing trainer using the algorithm in 1 and camera setup and configuration in 2
4. Determine the accuracy of the touch typing trainer in finger-key identification
5. Develop a new keyboard typing metric that uses both the key and finger used to press as parameters

1.4 Scope and Limitation

This research will focus on typing on a 60% keyboard. Figure 1.1 illustrates this type of keyboard. This type of keyboard only has the alphanumeric part of the keyboard. This limits the number of keys to be checked and the expected movement of the hand. Furthermore, the keycaps will also be of a light color, while the surface that the keyboard rests upon will be of a dark color.

In addition, the keyboard layout will be American National Standards Institute (ANSI). This layout is described by the American National Standards Institute (ANSI INCITS 154-1988, 1999).

This is the most common layout in the United States. However, it is also used in numerous English-speaking countries such as the Philippines, Malaysia, and India (Apple, 2021).

The setup that will be captured will be simplified and fixed for the development and testing of the finger-key identification submodule. This includes the keyboard, surface it is placed on, and lighting of the entire area. Furthermore, the setup will not be tested by other respondents, but by the researchers only — as the focus is on the development and testing of the finger-key identification submodule and not on the touch typing trainer and its effects on typing.

The program will expect the that user has all ten digits and has no hand, finger, or wrist deformities. In addition, only the placement of the hands, fingers, and wrists will be taken into account when determining if the ergonomics of the user while typing is healthy. The program will not check seating position, angle of elbows, and other metrics for an ergonomic typing posture while typing.

Capturing of the video to be analyzed by the program would be limited to a single 720p webcam that is capturing in 30 frames per second. The camera will be pointed downwards facing the keyboard and the hand.

Figure 1.1: A 60% keyboard in ANSI layout. Reprinted from Matt3o. (2014). Filcom MINILA Air pictured with Logitech M705 mouse for scale. Retrieved October 28, 2021, from https://deskthority.net/wiki/File:Filco_MINILA_Air.jpg

1.5 Significance of the Research

This research is beneficial for all users of physical keyboards. These include a vast majority of the population as there are a lot of professions that heavily rely on keyboards. Examples include developers, physicians, educators, and accountants. By having better ergonomics while typing — by touch typing correctly — wrist injuries can be prevented, and typing speed may be increased

This research also helps educators, especially early educators teaching beginner typists. By automatically checking for ergonomics, posture, and correct technique, the burden of checking each student is lessened, and directed interventions for bad habits can be easily created as students with these bad habits are easily identified

This research has a direct impact on people that has hand or wrist injuries that are caused by poor typing habits. By correcting these poor habits, pain from these injuries will be lessened, and even be prevented from occurring in the first place. A specific example of this is by reducing ulnar deviation which affects the nerve that is indicative of CTS (Toosi et al., 2015).

Chapter 2

Review of Related Literature

2.1 Keyboard Typing

Keyboard typing is the process of using a keyboard to input characters in a system. In the context of this paper, keyboard typing will refer to the act of using a physical keyboard to input characters in a computer system.

2.1.1 Keyboard Layouts and Form Factors

One key characteristic of a keyboard is its physical attributes. Keyboards come in a lot of layouts and form factors. Keyboard layouts are the shapes, size, and positions of a key on a keyboard while the form factor of a keyboard refers to its shape and dimensions. The form factor also refers to the number of keys included in the keyboard (Parkkinen, 2018). By combining different layouts and form factors, different permutations of a keyboard can be created.

Different keyboard layouts and form factors also produce different effects for the user. This is due to how vastly different some keyboard layouts and form factors are from one another. Some layouts focus on ergonomics, while others focus on typing speed. Some form factors were designed for aesthetics, while others focus on comfort and health. As such, different layouts may affect typing performance, ergonomics, and long-term health effects (Ciobanu et al., 2016).

ANSI and ISO Layout

There are two common keyboard layouts around the world — International Organization for Standardization (ISO) and ANSI.

ANSI INCITS 154-1988 is the standard that first defined the ANSI layout. Figure 2.1 illustrates what the ANSI layout looks like. This layout is also used by countries other than America. Examples of countries that use this layout as its standard is the Philippines, China, and Korea (Apple, 2021). However, these countries also opt to modify the layout by adding extra layers to accommodate other character sets.

ISO/IEC 9995-1:2009 is the standard series that defines a framework that is used to create other layouts. Layouts created from this standard are colloquially called ISO Layouts. Countries around the world use this framework to create layouts that fit the characters in their language. Examples of countries that use this framework to create their own layout are France, Greece, Canada, and Sweden (Apple, 2021).

Both of these layouts usually utilize the same key ordering. This ordering is commonly called QWERTY, based on the first five characters of the first row of this specific layout.

There are other layouts available, however, they are not as common as the two previously mentioned layouts. Examples of this include Japanese International Standards (JIS). Other esoteric layouts, like Tsangan or split-backspace, also exist. These layouts modify the ISO and ANSI standards by adding or removing certain keys to fit the character set of a language, or for additional keys. Other layouts are also exactly the same as ANSI or ISO, however, these layouts change the arrangement of the alphabet within the keyboard.

Despite the ubiquity of these common layouts, studies have shown that these layouts are not ergonomic. The main issue with these layouts is the random configurations of the letters. The randomness of the layout necessitates memorization of the layout which reduces the ease of learning, reduces performance in typing by reducing speed, and increases of typing errors (Ciobanu et al., 2016).

Keyboard Form Factors

There is only one common keyboard form factor used worldwide: the full-size keyboard. This keyboard contains all the keys specified in the keyboard layout. This includes the alphanumeric keys, the function keys, the navigation cluster, and the numpad.

Other common keyboard form factors are based on the full-size keyboard. The name of these layouts, 60%, 75%, and 80% reference the remaining number of keys after cutting a portion off from the full-size keyboard. The 60% keyboards only contain the alphanumeric cluster while the 80% and 75% layouts retain the navigation cluster and the function keys (Parkkinen, 2018). The main draw for using keyboards with reduced sizes is for aesthetics, space constraints, and ergonomics.

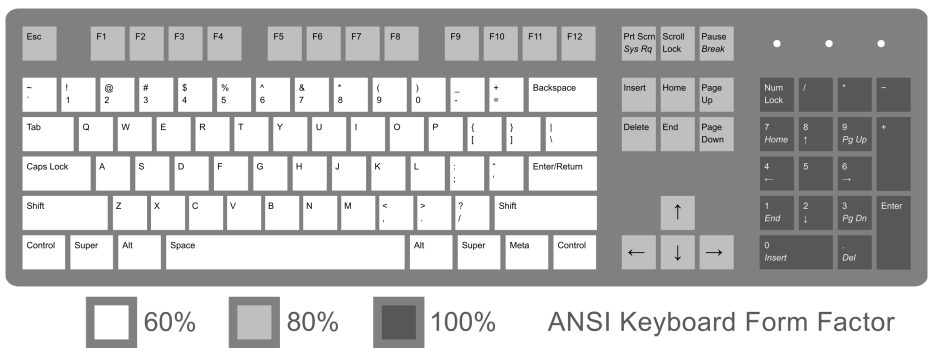


Figure 2.1: ANSI Keyboard layout with form factors. Reprinted from Rumudiez. (2013). Correctly labeled modifier keys for the ANSI Keyboard layout. Retrieved October 18, 2021, from https://commons.wikimedia.org/wiki/File:ANSI_Keyboard_Layout_Diagram_with_Form_Factor.svg

Ergonomic Keyboards Layouts and Form Factors

There have been other keyboard layouts and form factors created to mitigate common issues associated with QWERTY layouts. These include Colemak, Dvorak, and Alphabetical layouts. However, studies have shown that the layout itself does not matter as beginners do not necessarily see the keyboard as a structured set, but rather as a random collection of characters, even if it is alphabetized (Norman & Fisher, 1982),

A different form factor has a great effect on ergonomics. One such example of a form factor is an ergonomic keyboard developed by Microsoft called Microsoft Natural MultiMedia Keyboard. Ripat et al. used this keyboard in determining that ergonomic keyboards can help in reducing symptoms of Work-related neck and upper limb disorders (WRNULD). Figure 2.2 shows the layout of the Microsoft Natural MultiMedia Keyboard.

Figure 2.2: Microsoft Natural MultiMedia Keyboard. Reprinted from DraugTheWhopper. (2014). MS Natural Multimedia Keyboard. Retrieved October 28, 2021, from https://commons.wikimedia.org/wiki/File:MS_Natural_Multimedia_Keyboard.png

There are other form factors other than the full-size keyboard and variations thereof that focus on ergonomics. One such example is a split keyboard layout where the keyboard is split in half, one for the left hand and one for the right. One such benefit, according to Ergodox EZ, is a more relaxed position due to typing at shoulder width.

2.1.2 Keyboard Typing Metrics

There are numerous metrics used to quantify keyboard typing performance. Two common metrics used in the majority of typing tests include Accuracy and Speed.

Standardized Keyboard Typing Assessments

To be able to measure these metrics, a keyboard typing assessment needs to be done. However, there are no standardized keyboard typing assessments (Donica et al., 2018). As such, teaching methods and assessments, like Keyboarding without Tears, Monkeytype, and Keybr, may produce different metrics for the same typist due to their difference in conducting the assessment.

Speed

Speed, also called as entry rate by Arif and Stuerzlinger, measures the number of characters entered in a specific time frame. The most common metric that measures speed is WPM. WPM as defined by Arif and Stuerzlinger is:

$$WPM = \frac{|T| - 1}{S} \cdot 60 \cdot \frac{1}{5} \tag{2.1}$$

Where, $|T|$ is the length of the text, S is the time in seconds spent writing the text. This time starts directly after the first character has been pressed, and ends when the last letter has been entered. As such, 1 is subtracted from $|T|$, as the time spent to find and press the first character cannot be accurately determined. However, some typing assessments do not subtract 1 from $|T|$. 60 refers to the number of seconds in a minute and $\frac{1}{5}$ normalizes the metric for the average length of words.

Other metrics also measure speed, but they aren't as commonly used as WPM. These include Characters per Minute, Gestures per Second, Adjusted Words per Minute, and Keystrokes per Second

Accuracy

Accuracy measures the number of correctly pressed characters in an input string. Accuracy, as defined by Bartnik, is:

$$ACC = \frac{|C|}{|T|} \cdot 100\% \tag{2.2}$$

Where $|C|$ is the number of correct characters and $|T|$ is the length of the text.

The inverse of accuracy is error rate, where the number of incorrectly pressed characters is measured instead. Arif and Stuerzlinger describe 5 common error rate metrics: Error Rate, Minimum String Distance Error Rate, Keystroke per Character, Erroneous Keystroke Error Rate, and Total Error Rate.

Limitations of the Metrics

These metrics are all based on the inputted characters by the user. These metrics do not take into account other aspects of keyboard typing such as posture, hand and wrist positions, and finger placement. Consequently, these metrics do not give a full picture of the performance of the person typing, and they only provide a cursory view of how a person types.

2.1.3 Keyboard Typing Methodology

Keyboard typing can be accomplished in numerous ways. The main difference between the different methodologies is the number of fingers used when typing and how the typist navigates the keyboard to find the keys. The methodology ranges from Hunt and Peck to Touch Typing, with variations of the two in between.

Hunt and Peck uses one finger on one hand to press a key. This method is aided by using vision to locate the specific key to press (Hoot, 1986). On the other hand, Touch typing uses standard QWERTY mapping to type without using visual cues. (Dobson, 2009) This mapping involves assigning certain fingers to certain keys. Figure 2.3 is the standard QWERTY mapping used for an ANSI layout. Kinesthesis and proprioception are used in locating the keys (Logan et al., 2016).

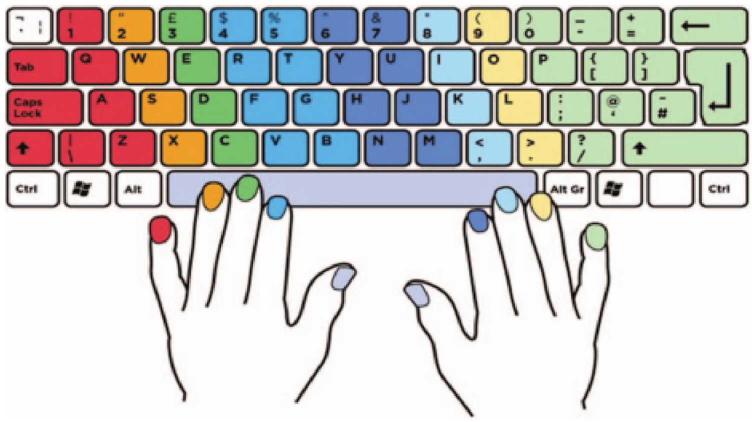


Figure 2.3: Standard QWERTY mapping for ANSI. Reprinted from Logan, G., Ulrich, J., & Lindsey, D. (2016). Different (key)strokes for different folks: How standard and nonstandard typists balance Fitts' law and Hick's law. *Journal of Experimental Psychology: Human Perception and Performance*, *42*(12), 2084–2102. https://doi.org/10.1037/xhp0000272

2.2 Keyboard Typing in Education

Today, students are expected to type essays, articles, and other submissions using word processors (Poole & Preciado, 2016). Testing is also commonly done using computerized assessments which require the need for keyboards (UMass Amherst, n.d.). As such, there is a need for students to be well versed in keyboard typing and for keyboard typing to be part of the curriculum.

Keyboard typing has been a part of this curriculum for a long time, with studies about effective methods to teach keyboard typing reaching as far back as 1986 (Hoot, 1986). Studies have continued to this day to continue to optimize and improve methods of teaching keyboard typing to students.

These studies start teaching kids in the kindergarten level and the studies try to optimize the teaching methods to improve the speed and accuracy of typing of the learners. By starting to teach touch typing to students early, these students will develop the potential for higher-level keyboard typing (Donica et al., 2018).

2.2.1 Expectations of Keyboard Proficiency

In the United States, keyboard typing is an expected learning outcome for third grade in the Common Core State Standards (Common Core State Standards Initiative (CCSS), 2010). At this grade level, only basic keyboard typing skills are required. By fourth grade, students are expected to have enough proficiency to type one page in one sitting. This is increased to two pages by fifth grade.

In the Philippine context, the Department of Education expects learners with a mental age of 4–6.9 years old to use correct posture and locate characters, learners with a mental age of 7–11.9 are introduced to home row finger placement, and learners with a mental age of 12 and above are expected to "use proper typing technique with efficiency and accuracy without looking at the keyboard" (Department of Education, 2020).

2.2.2 Current Teaching Methods

Current teaching methods involve replicating a given text. Learners then copy the text into a given text field that records the typed characters. Correct and incorrect characters are then identified, and suitable errors are presented. Afterward, metrics, such as WPM, and accuracy are given ("About TypeRacer," 2021; Bartnik, 2021).

Through this process, the learner goes through the three stages of Motor Learning Theory. The student undergoes the cognitive stage where they try to understand and create strategies to accomplish the given task. Then the associative stage follows where the strategies and skills learned from the previous stage are refined. At this stage, the learners are expected to rely less on visuals to locate the keys and more on kinesthesis. By the final stage, the autonomous stage, the learner does not rely on visuals at all and focuses on using kinesthetic feedback to find the keys. By this point, the learner has progressed from using Hunt and Peck, to becoming proficient in touch typing. (Donica et al., 2018)

Keyboarding without Tears

Keyboarding without Tears is a web-based application and curriculum that teaches students touch typing. However, one key differentiator of this curriculum is the usage of a row-based standard mapping, rather than a column-based standard mapping that is common in other teaching guides. Figure 2.4 shows the standard mapping used in this curriculum.

This curriculum is self-directed and learners can learn at their own pace. At its core, the curriculum is designed to be 36-week long with 5–10 minutes of lessons per day. The lessons in the curriculum follow the three stages of Motor Learning Theory ("Keyboarding Without Tears — K-5," 2020).

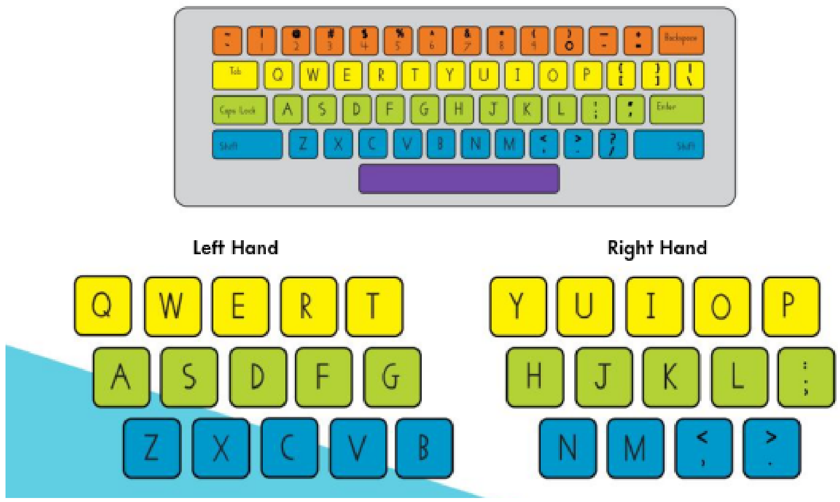


Figure 2.4: Row based standard mapping. Reprinted from Keyboarding Without Tears — K-5. (2020). Retrieved October 26, 2021, from https://issuu.com/handwritingwithouttears/docs/kwtbrochure2020/1

Monkeytype, Typeracer

These two keyboard typing tests are similar. They follow a common experience where users type a predetermined phrase, quote, or random words, and metrics are given after the test. Afterward, the learners may try the test again, or choose another set of words to type. These typing tests do not have a structured curriculum for learning how to touch type. It is left to the learner to practice and learn on their own ("About TypeRacer," 2021; Bartnik, 2021).

Keybr

Keybr is similar to Monkeytype and Typerace, in that they also have the users type a predetermined phrase, quote, or random words. However, this application has more guidance compared to the two. Keybr uses statistics to create typing lessons that are appropriate to the current typing proficiency of the learner. The words selected are random at first, and the skill level of the learner is determined by the performance of the user with these words and characters. The information gathered is then used to generate new words for the next iteration. As an example, if a learner has difficulty in typing the letter q, the next iterations will have a lot of words that contain the letter q.

Statistics from their website show that this learning method is successful, with some learners improving their typing speed by 20–40 WPM ("keybr.com - Typing lessons," 2021).

2.3 Keyboard Typing in Health

There have been a lot of studies that show the effect of keyboard typing, and its associated movements (or lack thereof), has an effect on the human body. These studies have shown that keyboard typing has an effect on our neck, shoulder, upper limb, wrist, arms, and fingers (Baker, Cham, Hale, et al., 2007; Szeto et al., 2005)

2.3.1 Health Issues arising from Keyboard Typing

WRNULD are a common issue that is associated with an elongated length of time maintaining a static posture. When using the computer, the posture commonly adapted by users has the neck and shoulder regions in a static hold for a long time. This results in forward neck flexion and increased muscle tension (Szeto et al., 2005).

In addition, it has been shown that 22% of computer users sustain musculoskeletal disorders of the upper extremity. This includes the neck, shoulder, hands, and wrists. (Gerr et al., 2002)

Carpal tunnel syndrome is also a common issue in the general population. This is caused by the chronic compression of the median nerve. There is a common belief that typing is one main cause for the disorder (Operations, 2021). There are no definite conclusions if this myth is true, however, a study by Toosi et al. found that typing causes ulnar deviation, especially if done without proper form. This ulnar deviation contributes to the swelling of the median nerve during and after typing. However, the authors noted that it is unclear if this swelling leads to long-term nerve injury.

2.3.2 Finger and Wrist Kinematics

The way people move their hands, wrists, and fingers differ between each person. This can be attributed to the different typing styles each person has. One key difference between people is the angle of the fifth digit.

However, there are some common movements and positions regardless of typing style: flexion, or the curving of the fingers, across the fingers, is decreasing across the hand, with the 2nd digit having the least flexion. This may be due to the instinct to reduce pronation of the hand, which in turn increases the distance of the 2nd digit to the keyboard. In addition, some people isolate or extend one of their thumbs, usually the one not used for pressing a key. This is also true for some people that do not use their fifth digit during typing (Baker, Cham, Cidboy, et al., 2007).

The movement and angle of the wrists also depend on the typing style of the typists. Some people do not reposition their hands, while others do. This difference comes from the way these people reach for certain far-away keys. Some stretch their fingers to reach far-away keys, while others move their entire hand to reach these keys.

For those that reach their keys by stretching their fingers, there is an increased probability that the wrists and fingers adapt non-neutral postures. These include wrist extension, ulnar deviation, and pronation, which may cause musculoskeletal disorders of the upper extremity (Marklin et al., 1999 as cited in Baker, Cham, Cidboy, et al., 2007)

2.4 Finger and Hand Tracking

Finger and Hand tracking is a method of tracking fingers and hands in 3D space using motion capture systems or computer vision. This technique allows computers to perform actions and analyzes on the motions and positions of these body parts.

2.4.1 Types of Tracking

Hardware Aided Solutions

Motion Capture Systems allow for capturing detailed skeletal motion in humans. These systems usually capture full-body motion, focusing on large parts of the human body, such as the torso, limbs, and head.

However, motion capture systems have difficulty in tracking more articulated body parts — with the fingers being one of them. The industry standard for capturing finger movements is through the use of an optical marker-based motion capture system. This is due to its ability to capture natural motion accurately.

This method uses cameras to triangulate the 3D location of markers attached to the limbs of a person. For finger tracking, 13–20 markers are placed on the fingers, and cameras are brought closer to track the small movements of the finger (Wheatland et al., 2015).

But this method is cost-prohibitive, and cannot handle occlusions well. Alexanderson et al. present a method for an optical marker-based motion capture system that can predictably recover from self-occlusion and has a better performance compared to previously used algorithms, however, the issue of cost and self-occlusion still persists.

Bend-sensor gloves are also an option for finger tracking. These gloves have sensors within them that track joint angles in the hand and fingers. One key differentiator of this solution compared to the others is the removal of self-occlusion in the data. As such, this is commonly used in sign language, and gesture recognition due to its accuracy.

However, these gloves need a lot of time to calibrate as cross-coupling of the sensors proves a problem. Cross-coupling is prevalent because the movement of one finger also moves other parts of the hand. These movements may cause a sensor aimed to track a specific movement of a different part of the hand to inadvertently detect a movement when there should be none (Wheatland et al., 2015).

Computer Vision

At its core, Computer Vision aims to perform tasks that the human visual system can do (Huang, 1996). This includes object classification, tracking, and gesture recognition, and face recognition. At the present, most computer vision systems utilize deep learning algorithms, and convolutional networks to gather information from an image, or a set of images. One such example of a convolutional network used in computer vision is Inception by Szegedy et al. which proposes a convolutional neural network architecture for object classification and detection.

2.4.2 Available CV Solutions for Tracking

OpenCV

OpenCV is an open-source computer vision and machine learning software library that houses ≈ 2500 optimized algorithms. This library is widely used by companies, researchers, and open source communities that utilize computer vision and machine learning in their projects. Examples of companies that use OpenCV include Google, Sony, and Honda.

The library has C++, Python, Java, and Matlab interfaces. The library also supports Windows, Linux, Android, and macOS, allowing for great developer experience, and wide deployment capabilities (Bradski, 2000).

MediaPipe

MediaPipe is an open-source computer vision framework that allows developers to create a perception pipeline. This perception pipeline is a directed graph of calculators. Data passes through the graph as packets and a group of packets constitutes a data stream. As the data passes through the pipeline, the calculators, produce the desired output.

This framework allows for performant object detection, hand and finger tracking, human pose detection. The framework also allows for combining multiple features, by adding them to the graph as calculators. MediaPipe has C++, Python, JS, and Coral interfaces. It also supports Android and iOS devices (Lugaresi et al., 2019).

MATLAB

MATLAB is a programming platform for the analysis and designing of systems. MATLAB is commonly used by engineers and scientists for computational mathematics (MathWorks, 2021b).

A toolbox offered by MATLAB is the Computer Vision Toolbox that contains algorithms, and functions for use in the development of computer vision, 3D vision, and video processing systems. By using the available algorithms in the toolbox, such as YOLOv2, and ACF, hand detection and gesture recognition is made possible in the platform (MathWorks, 2021a).

2.4.3 Applications

There have been multiple applications and products that utilize hand and finger tracking as their main component.

Dorfmuller-Ulhaas and Schmalstieg, 2001 presents a use case for finger tracking in augmented environments. In the paper, interaction in a virtual environment through the use of gestures. The tracking system uses an optical marker-based motion capture system where the user wears a glove with retroreflective markers.

Hsu et al., 2014 used a Kinect, a 3D sensing device by Microsoft that uses depth data, to track fingers to play virtual instruments. Virtual Pianos and Guitars were created and played with reliable and stable tracking.

Yousaf and Habib, 2014 created a virtual keyboard that operates using finger tracking. The tracking uses the movement of the finger joints as the basis for selecting which key to press. A camera captures the movement, and the resulting video stream is used for hand region detection and finger joint localization. Using probabilistic regional density-based kernel tracking, finger joint trajectories are gathered. Feature vectors are then interpreted from the trajectories. These feature vectors are used in logic-based techniques and Dynamic Bayesian Network for classification, detection, and recognition of keystrokes.

2.5 Summary of the Research Gap

While there are a lot of applications and curriculum aimed at teaching touch typing, there is no automated system available that detects if a person uses the correct finger to press a key.

By having this system, educators can accurately determine if and when a student is having a hard time typing and if these students will need an intervention to correct mistakes.

This is also important because certain movements and hand positions will cause nerve and muscular disorders that will impact the user. By correcting these problematic movements and hand positions, these disorders can be prevented.

Chapter 3

Methodology

3.1 Setup

The experimental setup and configuration was composed of three elements: the camera, the keyboard, and the environment. Figure 3.1 is an image of the experimental setup

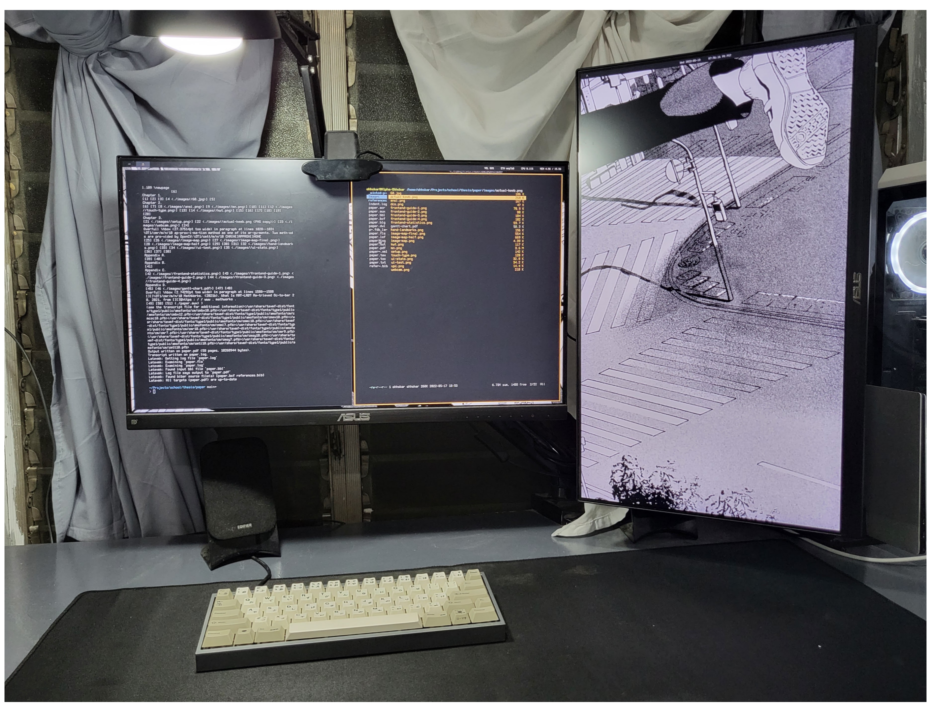


Figure 3.1: The experimental setup

3.1.1 Camera

The setup used a single monocular camera positioned in a top-down view. The camera captured the entirety of the keyboard and the movement of the ten fingers. To do so, it was mounted on top of the monitor with the camera pointed down towards the table.

Figure 3.2: The camera angled downwards to capture the keyboard

Figure 3.3: Image captured by the camera in one of the training iterations

The specific camera used was a Logitech C920. It is a 3 mega pixel webcam that is capable of capturing color video in 1080p/30fps and 720p/30fps with a diagonal field of view of 78°. This camera has a universal mounting clip that allows the camera to be correctly positioned within the experimental setup (“Logitech C920 PRO HD Webcam, 1080p Video with Stereo Audio,” n.d.).

Figure 3.4: Logitech C920. Reprinted from Logitech C920 PRO HD Webcam, 1080p Video with Stereo Audio. (n.d.). Retrieved January 26, 2022, from https://www.logitech.com/en-ph/products/webcams/c920-pro-hd-webcam.html

3.1.2 Keyboard

The keyboard used was a 60% keyboard as shown in Figure 3.3. This limited the necessary mapping for the algorithm to the alphanumeric portion of the keyboard. This keyboard choice also reduced the area that the camera captured, as this keyboard type is considerably smaller compared to a full-size keyboard. The keyboard also had its keycaps and case in a light color that contrasted the dark surface it was placed on. It also had a dark USB-C cable to blend with the dark surface. These color coordination steps improved initial keyboard detection. In addition, the keyboard layout was ANSI, due to the availability and widespread adoption of the layout in the Philippines.

3.1.3 Environment

The setup was lit with a lamp besides the camera. This light source evenly lit the keyboard and the fingers used for typing. The light source used was a common LED bulb rated at 9 Watts with 700 lumens. This light was white with a color temperature of 6500k.

In addition, the surface where the keyboard was placed on was solid black without any variation of color. This also improved initial keyboard detection.

3.2 Algorithm

3.2.1 Computer Vision based Keyboard Detection, and Key Mapping

A computer vision algorithm was created as a starting point mapping the keys of the keyboard within a video. A rough flowchart of the algorithm is shown in Figure 3.5.

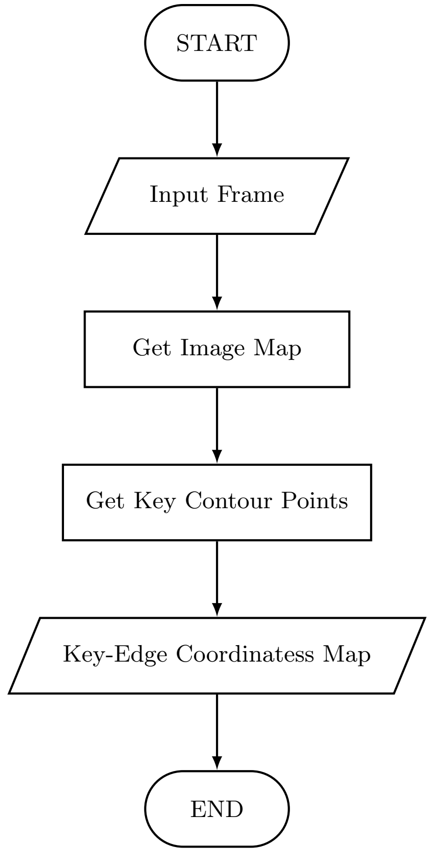


Figure 3.5: Flowchart of Keyboard Detection and Mapping Algorithm

Get Image Map

The first portion of the algorithm creates an image map that is overlaid over the detected edges of the keyboard. The flowchart for this function is shown in Figure 3.6.

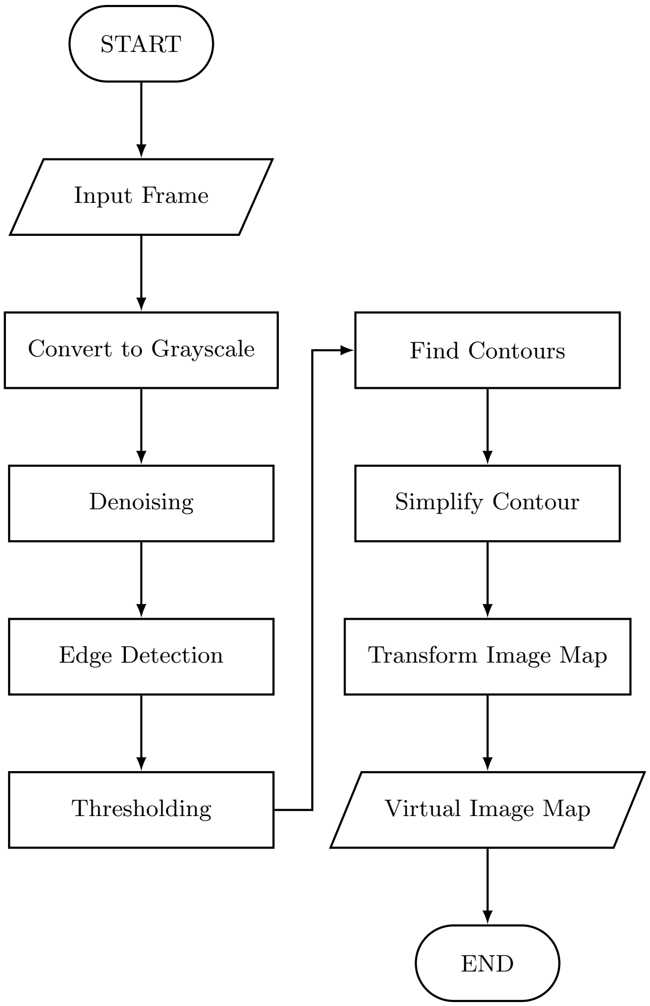


Figure 3.6: Flowchart of Get Image Map

Convert to Grayscale The input frame from the camera was converted to a 256 level grayscale image using `cvtColor` of OpenCV ("OpenCV: Color Space Conversions," n.d.). This step was performed because future steps of the algorithm did not require color values to work. In addition, this step optimized the algorithm as the number of dimensions analyzed was reduced.

Figure 3.7: Camera capture converted to grayscale

Denoising The algorithm denoised the grayscale image using a bilateral filter as implemented by OpenCV ("OpenCV: Image Filtering," n.d.). This filter takes the range of the image into account, rather than just the domain. This resulted in an image that is smoothed while preserving its edges (Fisher, n.d.).

Figure 3.8: Smoothed image with intact edges

Edge Detection The algorithm used a Sobel filter for edge detection. This filter uses a 3×3 kernel that is convolved twice. Once horizontally, and another vertically to produce a grayscale image of the outlines within the frame. The kernels used by the Sobel filter (Sobel, 2014) is shown in Figure 3.9.

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure 3.9: Sobel Operator Kernels. Reproduced from Sobel, I. (2014). An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*

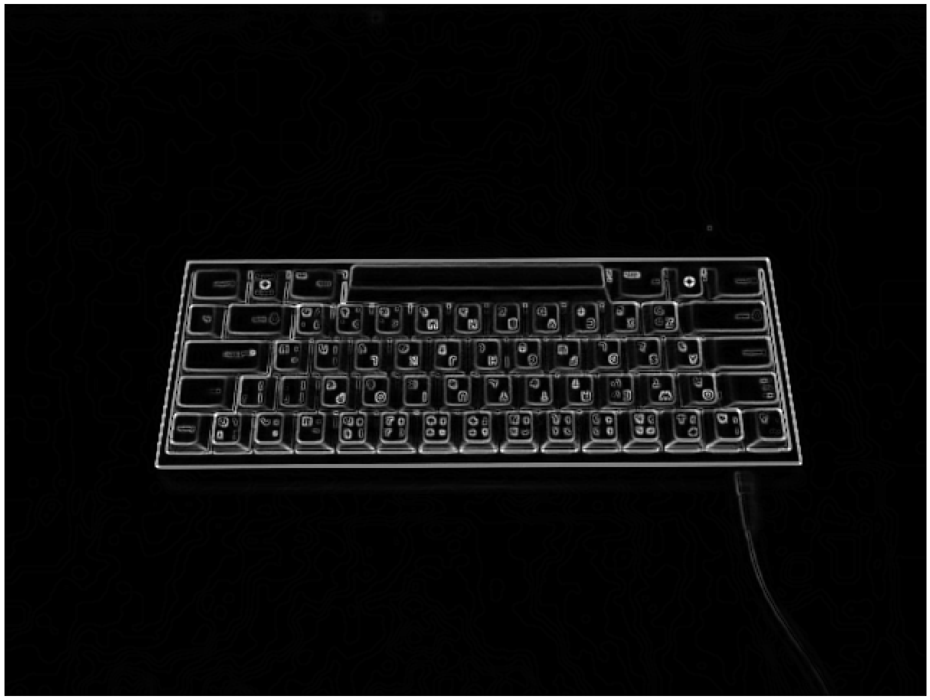


Figure 3.10: Output of the Sobel filter

Thresholding The output of the Sobel filter is a grayscale image of 255 values, with each gray pixel indicating edges within the image. There is a need to reduce the range of these values to improve the ability of the algorithm to find the contours of these edges. To do so, the algorithm utilized Otsu's algorithm to perform automated thresholding. Otsu's algorithm determines a single threshold that is most optimal for the image (Otsu, 1979). This outputs a black-and-white image, with only two values.

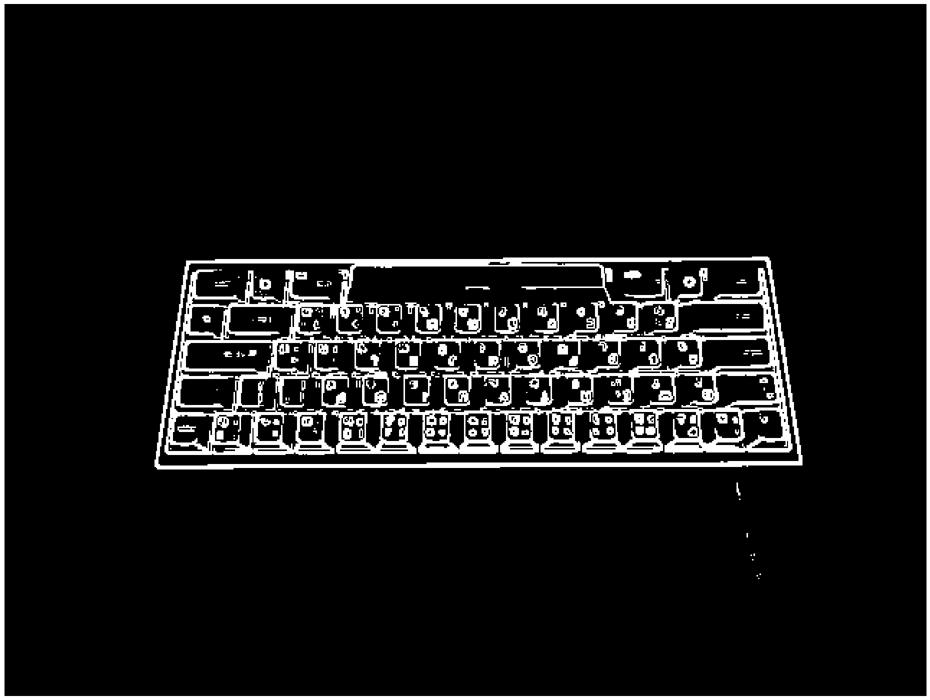


Figure 3.11: Threshed image of the edges

Find Contours Contours are curves joining all continuous points that have the same color or intensity ("OpenCV: Contours : Getting Started," n.d.). The OpenCV function `findContours`, with the `CHAIN_APPROX_SIMPLE` contour approximation method, was used to find the contours of the outlines of the object found using the previous step. This approximation method removed redundant points and returned the least amount of points that describes the shape. This OpenCV function implements the algorithm of Suzuki and Abe, 1985 in their paper "Topological structural analysis of digitized binary images by border following."

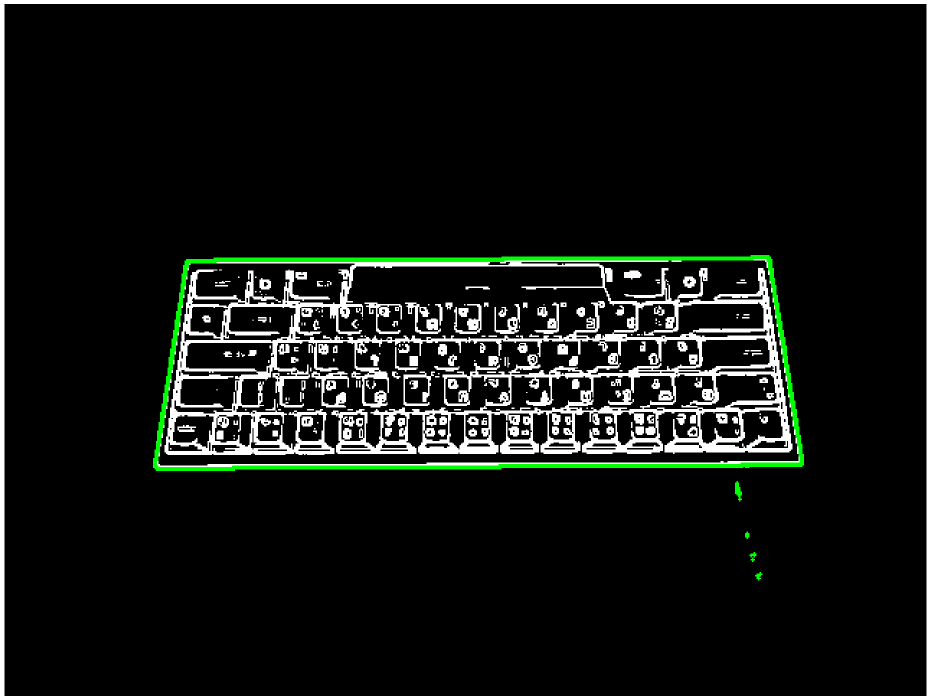


Figure 3.12: All the contours found in the image

Simplify Contour However, the points returned by the function can contain more than the four extreme points at the edges of the keyboard. In addition, more than one contour may be found within the image. As such, the algorithm sorted the contours by area, and selected the largest one.

After sorting, the algorithm performed the Douglas-Peucker algorithm for Line Simplification (Saalfeld, 1999). This reduced the number of points in the largest contour to the minimum amount. In ideal cases, the number of points would be four, with each point corresponding to the edges of the keyboard. However, there were times when other objects would be within the frame, or they intersected with the keyboard. This would result in a contour that would be defined by more than four points — which caused the entire algorithm to fail.

The contour points that described the contour were then sorted clockwise. This was a necessity since the algorithm of Suzuki and Abe, 1985 does not guarantee that the largest contour's points were returned in a clockwise arrangement, which is a requirement for the next step

Figure 3.13: Simplified contour

Transform Image Map The virtual keyboard map is a rectangular image that contains individual Region of Interest (ROI) for each key in a 60% ANSI keyboard. Each key has a corresponding color assigned to it and this color fills the region where this key is located at. Appendix Chapter A is a table that shows the hex color code and key mapping present in the image map.

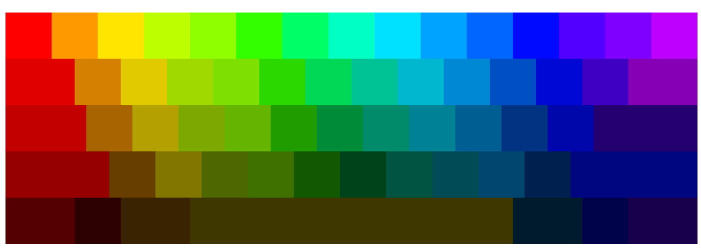


Figure 3.14: Initial Image Map

The virtual map was stretched over the keyboard using OpenCV's `warpPerspective` function. The function requires a perspective transform that was calculated using `getPerspectiveTransform`. This function takes in two arrays with four points each. The input points are the edges of the virtual map, ordered clockwise. The output points are the four points of the contour, ordered clockwise ("OpenCV: Geometric Image Transformations," n.d.). The resulting transform was then used by `warpPerspective` to apply the transform to the virtual map.

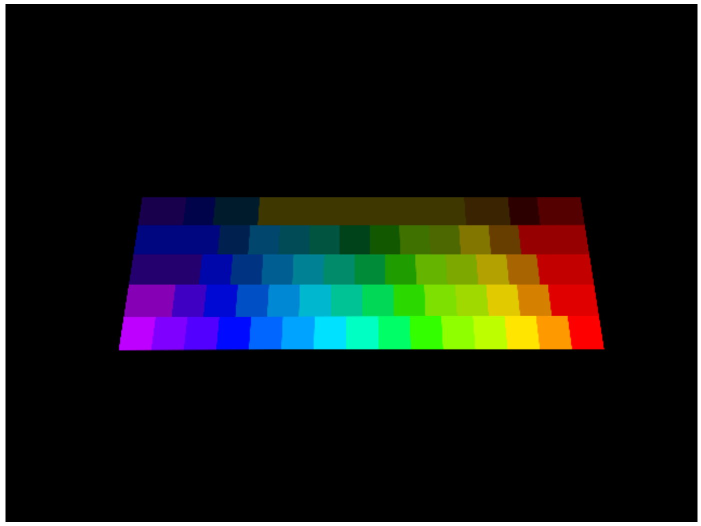


Figure 3.15: Transformed image map

Get Key Contour Points

The second portion of the algorithm pre-computes the contour points of each key in the keyboard based on the generated virtual map and the Key-Color Values map. The flowchart of this function is shown in Figure 3.16

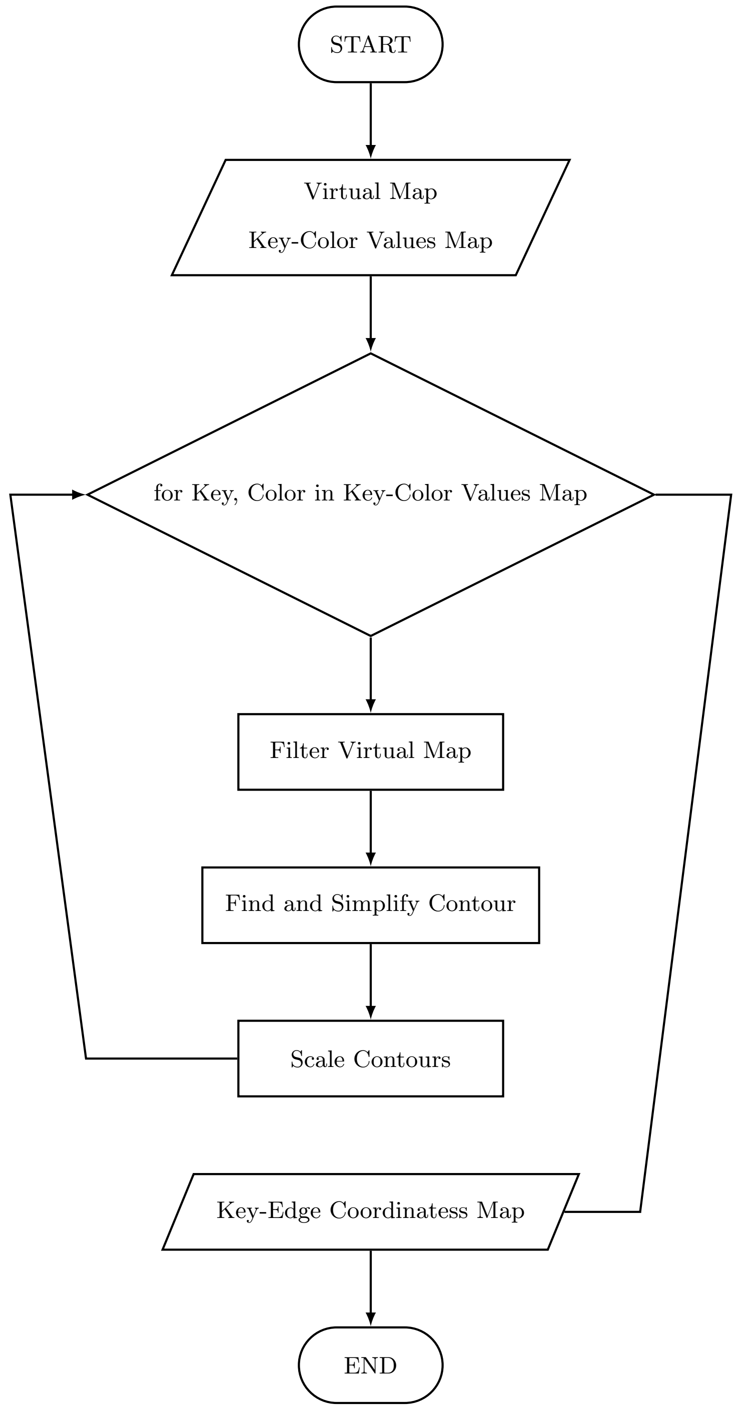


Figure 3.16: Flowchart of Get Key Contour Points

Filter Virtual Map The color that corresponds to each key in the virtual map is stored in the Key-Color Value Map. This color was used to create a mask for the virtual map for a specific key. This mask was obtained by using this snippet of code `(virtualMap == key).all(axis=2)`. This mask was then used to black out the rest of the virtual map, leaving only the ROI of the key in the image.

This specific method was used as the other method that was trialed, `np.where(virtualMap != key)`, did not consider all 3 channels when comparing each pixel — i.e. a key assigned with the color value of `[100, 0, 0]` will not be blacked out if the color value of the key to be isolated is `[100, 243, 0]` as the blue channel of the pixel, `100` is equal in both. This would result in other keys remaining in the virtual map, even if only one key corresponds to that color.

Figure 3.17: Filtered key

Find and Simplify Contours The same method in finding and simplifying the contour of the keyboard was used to find and simplify the contour of the ROI of the key.

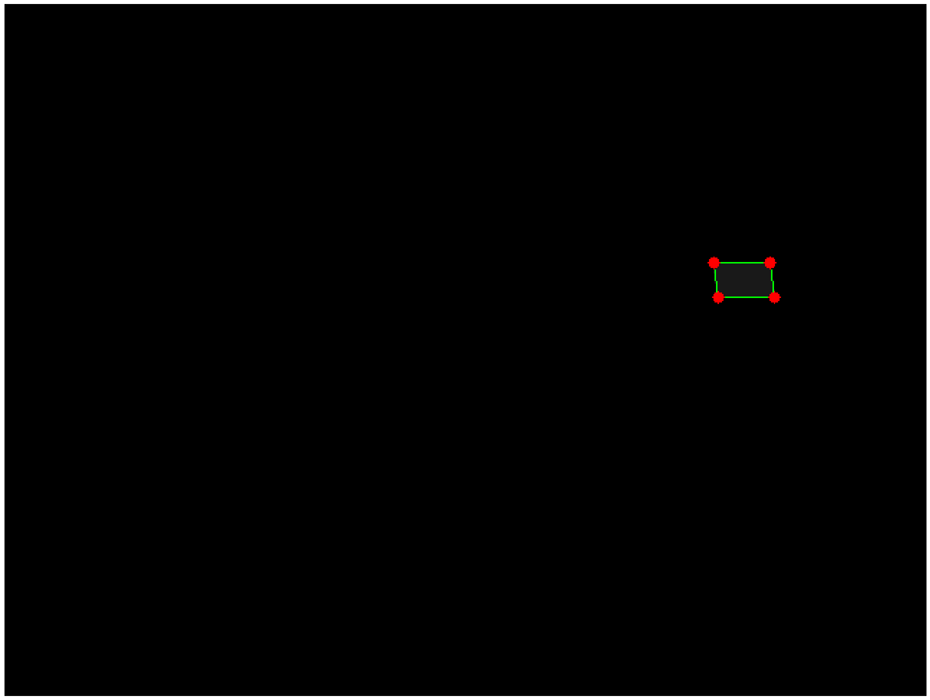


Figure 3.18: Contour points of the ROI

Scale Contours During testing using training data, a lot of the failures in finger-key identification was due to the tight fit of the contour to the key. The contour on its own did not give enough buffer for the placement of the finger. This buffer was needed as, in some instances, the finger used to press the key was not exactly on top of the key, but rather to its side. The algorithm accounts for this situation by adding a buffer. This was achieved through scaling the contour points by seven pixels on each side. This value was obtained after testing other possible values. A buffer of five pixels did not have a lot of effect in reducing the number of failures. A buffer of ten pixels did reduce the number of failures, but it also greatly increased the algorithm's uncertainty in determining the finger — uncertainty is defined as detecting two or more fingers within one ROI. Seven was a good middle ground in decreasing failures, without greatly increasing uncertainty. More information about this can be found in the Results and Discussion at Section **??**

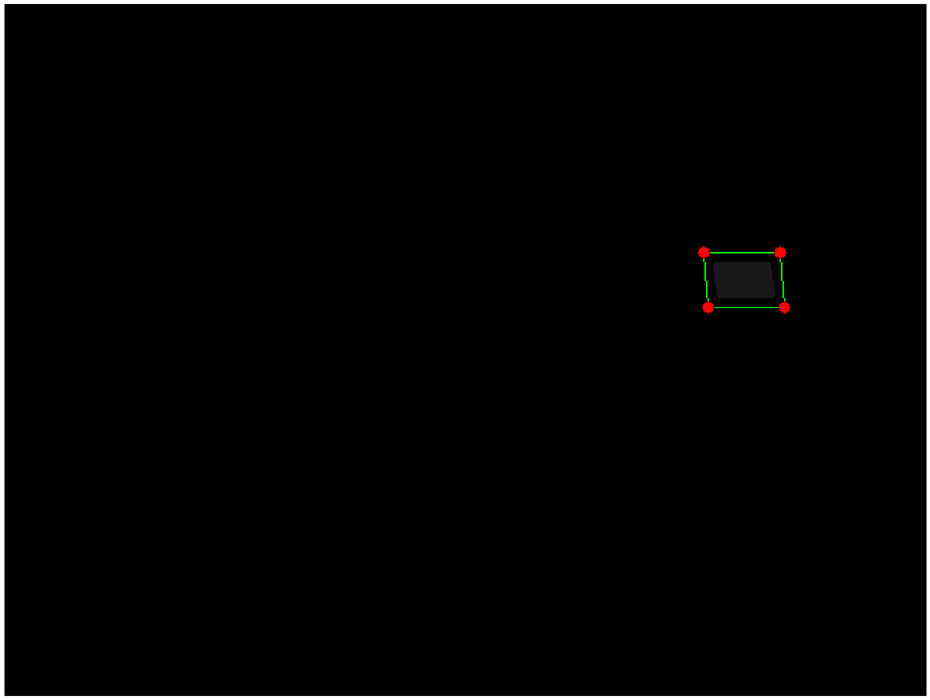


Figure 3.19: Contour points of the ROI

Key-Edge Coordinatess Map After all the keys have passed through the loop, a Key-Edge Coordinates Map was generated. Each key now has four points that serve as the coordinates of the edge of its ROI.

The coordinate system that the contour points are based on had its origin at the top left, starting at 0, 0. Going to the right increased the value of the x-axis, and going to the bottom increased the value of the y-axis. This results in a coordinate system that only has positive values with each value corresponding to a pixel.

3.2.2 Computer Vision based Finger Detection, and Tracking

MediaPipe Hands was used as the finger detection and tracking solution. This algorithm is composed of two ML models working in conjunction to be able to detect the different parts of the hands and track them accurately.

Palm Detection Model

The first model, the Palm Detection Model detects the initial hand locations using a single shot detector model based on the paper by Liu et al., 2016. This model achieves an average precision of 95.7% in palm detection (Lugaresi et al., n.d.).

MediaPipe Hands detects the palms first, instead of whole hands with one model because hands lack high contrast patterns. This reduces the model's ability to detect hands with accuracy. In addition, detecting a palm is simpler compared to detecting hands with articulated fingers since estimating a bounding box around a rigid object, i.e. a palm, is much simpler. Furthermore, a palm can be modeled using only square anchors reducing the number of anchors by a factor of 3–5 (Lugaresi et al., n.d.).

Hand Landmark Model

After the palms have been detected and an appropriate anchor has been established, the Hand Landmark Model pinpoints 21 3D hand-knuckle coordinates inside the detected hand. This is done using direct coordinate prediction.

This model was trained using 30,000 manually annotated, real-world images with 21 3D hand-knuckle coordinates. Using this information, the model can also accurately add landmarks to partially visible hands and hands with self-occlusion. This is also made possible by the model's consistent internal hand pose representation (Lugaresi et al., n.d.).

3.2.3 Integration for finger-key identification and mapping

The two previously chosen algorithms was combined to accomplish finger-key identification and mapping.

The integration of the algorithm was a two-step process. The first step was to get the Key-Edge Coordinates Map shown in Section 3.2.1.

The second step runs whenever a key press has been detected. This step used the Key-Edge

Coordinates Map in conjunction with the finger tracking algorithm selected in Section 3.2.2. The flowchart of the algorithm is shown in Figure 3.20

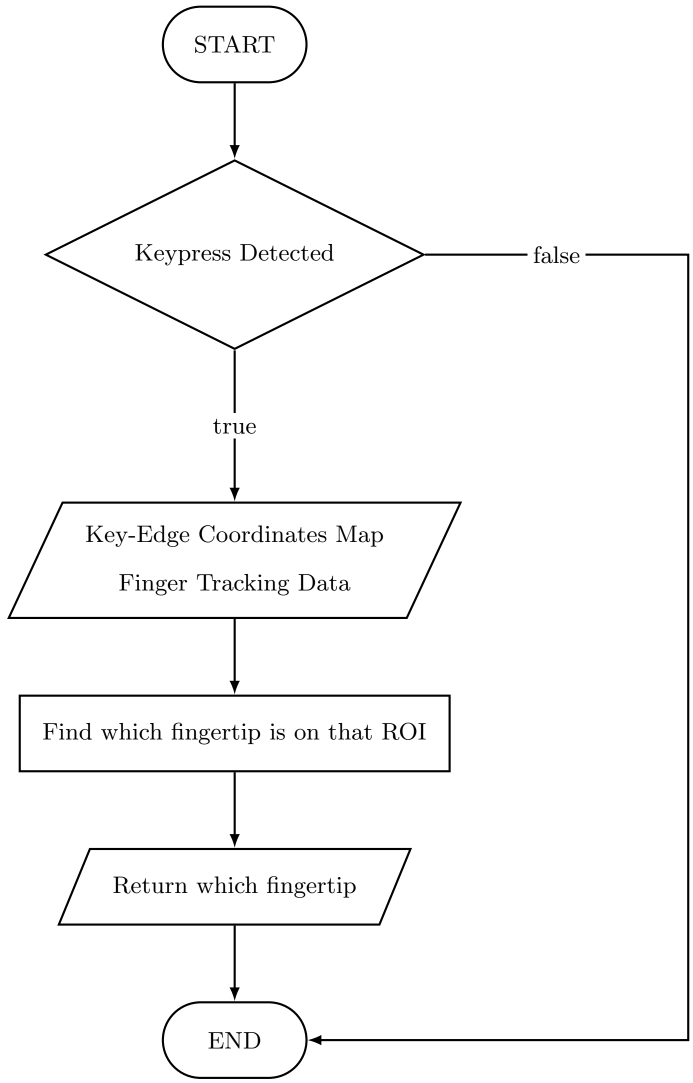


Figure 3.20: Flowchart of the overall flow

Find which fingertip is on that ROI

The Finger Tracking Data contained the pixel positions of each landmark of the hand. For this step, the algorithm finds the landmark which is positioned within the single ROI obtained from the previous step. This was done using a series of checks.

Each landmark's coordinates was compared to the coordinates of the edges of the ROI. A landmark was determined as positioned within the ROI if all the following conditions are true:

1. In the X axis, the landmark's coordinates is greater than one or both of the two coordinates found of the left side of the ROI
2. In the X axis, the landmark's coordinates is less than one or both of the two coordinates found of the right side of the ROI
3. In the Y axis, the landmark's coordinates is greater than one or both of the two coordinates found of the top side of the ROI
4. In the Y axis, the landmark's coordinates is less than one or both of the two coordinates found of the bottom side of the ROI

These conditions maximized the total area of the ROI, and it is not strict about exact accuracy. In essence, these conditions creates a perfectly rectangular box that contained the quadrilateral formed by the four points from the coordinates.

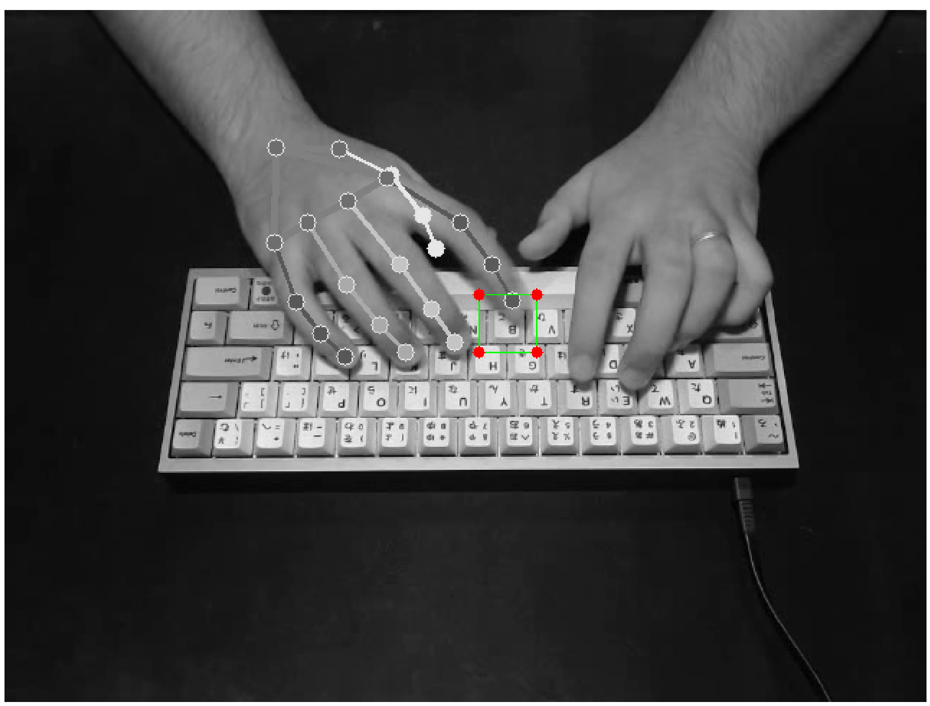


Figure 3.21: Finger tip within an ROI

Return which fingertip

The landmark was then used to determine which specific finger corresponds to the key press. Figure 3.22 shows each possible landmark that may be returned from the previous step. This step returned the name of the landmark, up until the first underscore. As an example, if the landmark found is `MIDDLE_FINGER_TIP`, this step will return `MIDDLE` denoting that the middle finger is the finger that corresponds with the key press. In addition, the specific hand will also be returned as one of two strings, `LEFT` and `RIGHT`, since this information is also bundled together with the landmarks. The two strings were then concatenated with an underscore. An example return value is `RIGHT_RING`.

As such, the expected output of the submodule is the hand and the finger used to press the key. This allows for a more flexible utilization of the data as it carries greater context, compared to just returning if the finger used was correct or not.

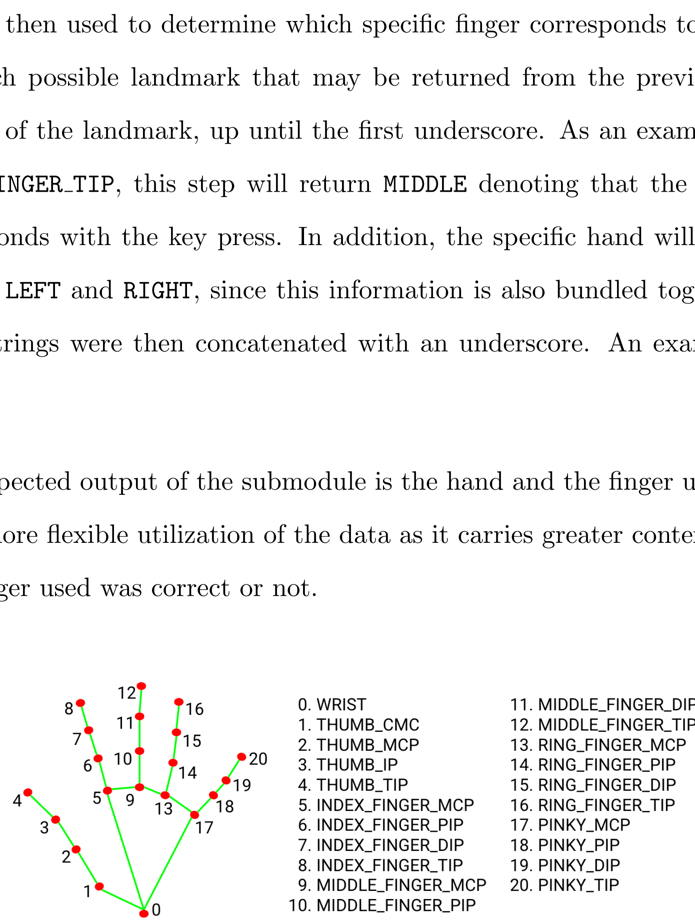


Figure 3.22: Hand landmarks that may be returned by the algorithm. Reproduced from Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (n.d.). Hands. Retrieved January 13, 2022, from https://google.github.io/mediapipe/solutions/hands.html

.

3.2.4 Implementation

The previously discussed algorithms was implemented using Python. The Keyboard Detection, and Mapping Algorithm in Section 3.2.1 was implemented as a separate submodule in Python using OpenCV. The Finger Detection, and Tracking Solution in Section 3.2.2 was consumed using the prebuilt Python package offered by the MediaPipe team. Finally, the integration of the two will be done as another separate submodule in Python.

3.3 Training and Testing

3.3.1 Typing Test Sequences

Typing test sequences are strings that was used in testing the user in their ability to type. The test sequences that was used came from text found in the public domain obtained from Project Gutenberg and the Internet Archive. Sentences were isolated from these text and used as test sequences if they fit the criteria.

The criteria for choosing test sequences were as follows: (1) $\approx$ 80% of the characters in the keyboard is present in a test sequence. (2) The number of words in a test sequence do not exceed 25. (3) Numbers and punctuations should be present in at least $\approx$ 30% of the total test sequences.

There was a total of 10 test sequences that was gathered. An example test sequence is "What of it, if some old hunks of a sea-captain orders me to get a broom and sweep down the decks?" from Moby Dick by Melville, 2001. The 9 other test sequences can be found in Appendix Chapter B.

3.3.2 Data Gathering

Video Capture

There were 3 groups of 10 videos that were captured by the researcher, for a total of 30 videos. Each group corresponds to three predetermined speeds of typing: slow (15wpm), average (35wpm), and fast (80wpm). These speeds were based on test data from "keybr.com - Typing lessons," 2021. The researcher typed the 10 typing test sequences at these predetermined speeds. During typing, errors were not consciously taken into account, and errors happened naturally as part of the typing process.

A Python script was created to facilitate this process. The script captured a video of the researcher as the researcher was typing the test sequences. Whenever the researcher pressed a key, the corresponding frame count was captured, and the key pressed was recorded in conjunction. This was stored in the format `FRAME_NO:KEY_PRESSED`, and each keypress was then stored as a line in a file.

Keypress Labelling

The researcher then manually perform finger-key identification for all key presses present in the video. This was also done through a Python script. The script parsed the file associated with a video. The frame number found at each line was then shown, and the associated data with the frame was superimposed over it. The researcher then pressed keys that corresponded with the finger used to press the key. This was then again stored in the format `FRAME_NO:KEY_PRESSED:FINGER_USED`, and each keypress was then stored as a line in a file.

Figure 3.23: Screen showing the labelling process

Test-Training Split

The annotated data was then divided into training and test data at a ratio of 60:40.

3.3.3 Training

A script was created to test the accuracy of the submodule. This script opened each video, and the associated labeled file. The script then parsed the labeled file, and opened the frame for each keypress. The finger-key identification submodule was then run, using the data needed for the submodule: the key pressed. The result of the submodule was then compared with the manual annotation of the keypress. The resulting data from the test was then stored in a `.csv` file with the following content:

Column Name	Definition
`file_name`	Name of the file
`frame`	Frame number
`key`	Key pressed
`finger`	Manual annotation of the finger used to press the key
`is_correct`	If the submodule was correct
`fingertips_detected`	The fingertips detected by the submodule
`keyboard_detection_time`	Time spent in creating the Key-Edge Coordinates Map
`finger_classification_time`	Time spent in classifying which finger pressed the key

Table 3.1: CSV File Contents

Adjustments were then made to the algorithm after each training pass to improve its performance. Some runs had improvements in its statistics without doing any adjustments. These were runs that showed errors in the manual annotation and were corrected afterwards.

3.3.4 Analysis

The resulting `.csv` file was imported into Google Sheets and key statistics were obtained based on the data. These statistics can be found on Table 3.2. These key statistics informed what to adjust in the algorithm and its overall performance.

Identifications	
Success	Number of successful finger-key identifications
Failed	Number of failed finger-key identifications
Success Percentage	Percentage of successful finger-key identifications
Failed Percentage	Percentage of failed finger-key identifications
Failure Types	
Mismatched Identification	Submodule output did not match manual annotation
No Identification	Submodule did not detect finger tips in ROI
Uncertain Successes	
Total	Total uncertain successes, defined as detecting two or more fingers within one ROI
Spacebar	All uncertain successes that were on the spacebar
Spacebar Percentage	Percentage of spacebar uncertain success over all uncertain successes
Non-spacebar	All other uncertain successes
Non-spacebar Percentage	Percentage of non-spacebar uncertain success over all uncertain successes
Non-spacebar over total Percentage	Percentage of non-spacebar uncertain success over all total keypresses
Average Running Time	
Keyboard Detection Time	Average keyboard detection time of all videos
Finger Identification Time	Average finger identification time of all key presses

Table 3.2: Acquired statistics

3.3.5 Testing

The same script was run through the test data. The same format of output was gathered, and the same key statistics were recorded.

3.4 Metric

There was a two total metrics obtained pertaining to finger and key mapping. The first was per test sequence, and the second was per key.

3.4.1 Per Test Sequence

The metric which calculates per test sequence is Finger Placement Accuracy (FP ACC). This metric computes the percentage of accurate keys pressed with the correct finger over the length of the typing test sequence. Inputting the wrong character, even if pressed with the correct finger, reduces FP ACC since FP ACC measures *accurate key presses*, and incorrect characters areconsidered as inaccurate key presses.

The equation for calculating FP ACC is as follows:

$$FPACC = \frac{|F|}{|T|} \cdot 100\% \tag{3.1}$$

Where $|F|$ refers to the number of accurate keys pressed with the correct finger and $|T|$ refers to the length of the text.

3.4.2 Per Key

This metric computes the Historical Finger Placement Accuracy (HFP ACC) for a certain key. This metric takes the times the user pressed the wrong key when asked for a certain key into account, the same as in FP ACC.This allows the user to easily verify which keys need attention and retraining if there is a high frequency of error.

The equation for calculating a key's HFP ACC is as follows:

$$HFPACC = \frac{|F_{char}|}{|C_{char}|} \cdot 100\% \tag{3.2}$$

Where $|F_{char}|$ refers to the number of accurate keys pressed with the correct finger for a certain character, identified as $char$. $|C_{char}|$ refers to the number of times the character has appeared in all test sequences for the user.

Chapter 4

Results and Discussion

Chapter 5

Conclusion

Chapter 6

Future Work

Chapter A
Key-Color Values Map

Hex Color Code	Corresponding Key
#FF0000	~
#FF9900	1
#FFE600	2
#BDFF00	3
#8FFF00	4
#33FF00	5
#00FF66	6
#00FFC2	7
#00E0FF	8
#00A3FF	9
#0066FF	0
#000AFF	-
#5200FF	+
#BD00FF	Backspace
#E00000	Tab
#D68000	Q
#E1CB00	W
#A0D900	E
#7EE000	R
#2BD900	T

Continued on next column

Hex Color Code	Corresponding Key
#00D856	Y
#00C395	U
#00B7D0	I
#0088D4	O
#004FC5	P
#0008D3	[
#3F00C5	]
#8600B5	|
#C30000	CapsLock
#A76400	A
#B2A100	S
#7CA800	D
#65B400	F
#1F9D00	G
#008B38	H
#018B6A	J
#008294	K
#005E93	L
#003482	:
#0007AA	”

Continued on next column

Hex Color Code	Corresponding Key
#23006D	Enter
#960000	LeftShift
#683E00	Z
#827500	X
#4D6800	C
#3F7100	V
#125800	B
#00421A	N
#005440	M
#004B55	,
#00466D	.
#002050	/
#000580	RightShift
#550000	LeftControl
#2D0000	LeftSuper
#3A2300	LeftAlt
#3E3700	Spacebar
#003238	RightAlt
#001C2C	RightMeta
#000349	RightSuper
#18004D	RightControl

Concluded

Chapter B

Typing Test Sequences

1. "What of it, if some old hunks of a sea-captain orders me to get a broom and sweep down the decks?" (Melville, 2001)
2. "I sat down on an old wooden settle, carved all over like a bench on the Battery." (Melville, 2001)
3. "I lay there dismally calculating that sixteen entire hours must elapse before I could hope for a resurrection." (Melville, 2001)
4. "How slowly the time passes here, encompassed as I am by frost and snow!" (Shelley, 1993)
5. "I listened to my father in silence and remained for some time incapable of offering any reply." (Shelley, 1993)
6. "Think you're escaping and run into yourself. Longest way round is the shortest way home." (Joyce, 2003)
7. "History, Stephen said, is a nightmare from which I am trying to awake" (Joyce, 2003)
8. "A man of genius makes no mistakes. His errors are volitional and are the portals of discovery." (Joyce, 2003)
9. "Never trust to general impressions, my boy, but concentrate yourself upon details" (Doyle, 1999)
10. "I have no data yet. It is a capital mistake to theorise before one has data." (Doyle, 1999)

Chapter C

Front-End Screens

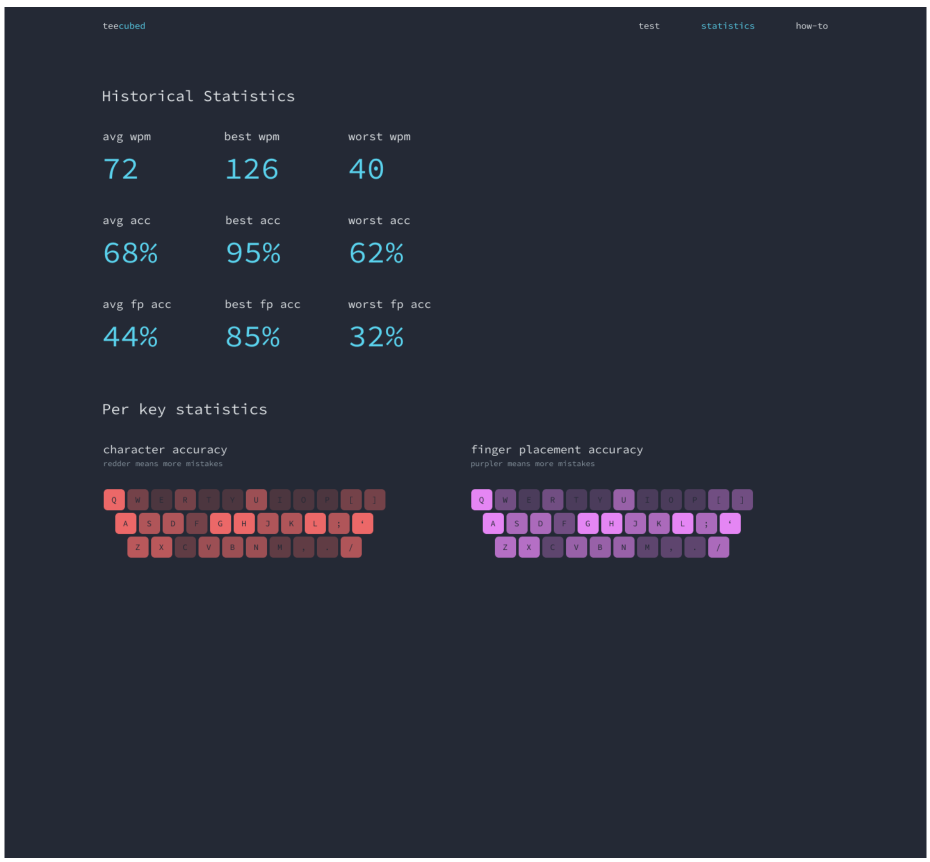


Figure C.1: Page showing historical statistics of the user

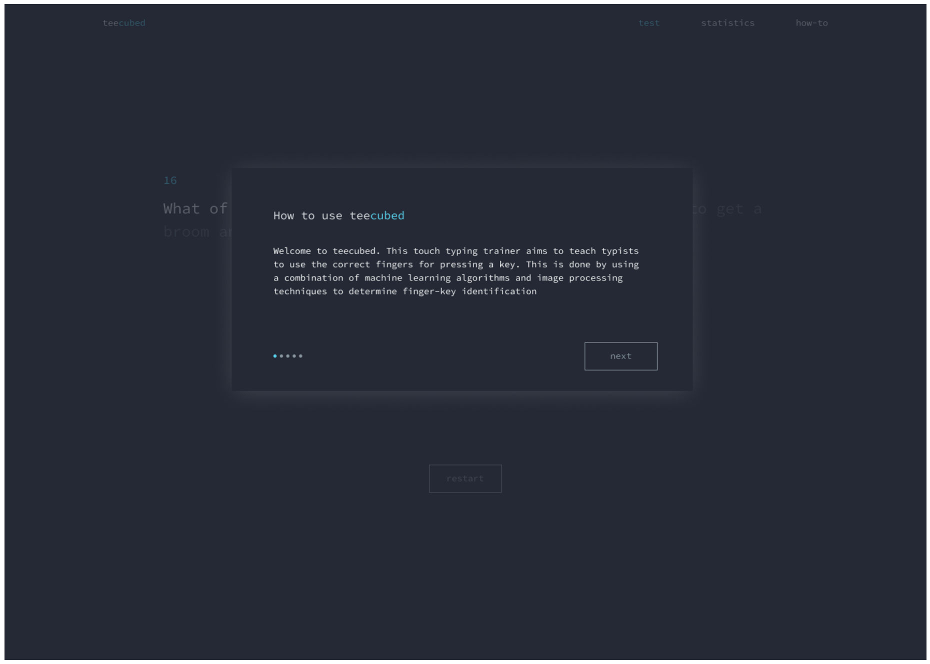


Figure C.2: Introduction to the platform during first open

Figure C.3: Step 1 of the tour of the platform

Figure C.4: Step 2 of the tour of the platform

Figure C.5: Step 3 of the tour of the platform

Chapter D

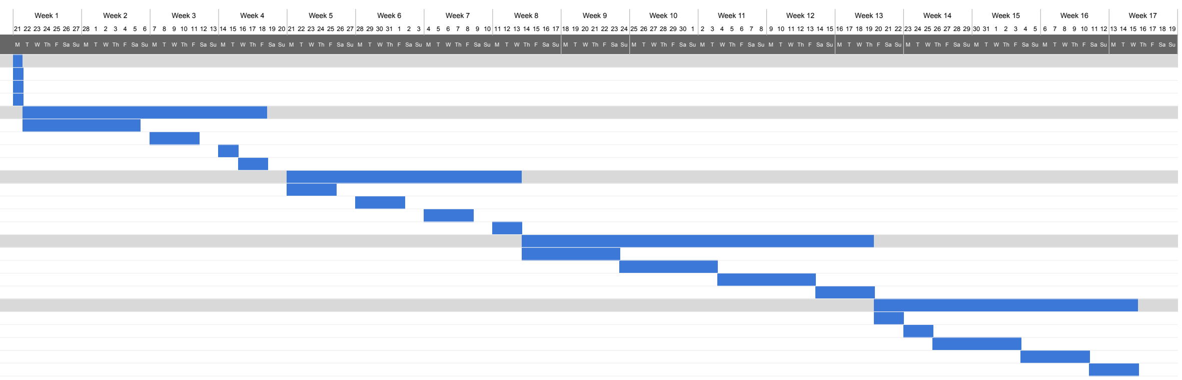
Gantt Chart

Development of a touch typing trainer with an emphasis on finger-key identification

Oscar Vian L. Valles

Project Start Date: 2/21/2022 (Monday)   Display Week: 1

WBS	Task	Start	End	Days	% Done	Work Days
1	**Experimental Setup**	Mon 2/21/22	Mon 2/21/22	1		1
1.1	Camera Setup and Placement	Mon 2/21/22	Mon 2/21/22	1	0%	1
1.2	Keyboard Setup and Placement	Mon 2/21/22	Mon 2/21/22	1	0%	1
1.3	Environmental Setup and Placement	Mon 2/21/22	Mon 2/21/22	1	0%	1
2	**Keyboard Detection and Mapping**	Tue 2/22/22	Fri 3/18/22	25		19
2.1	Convert to Greyscale and Edge Detection using Sobel Filter	Tue 2/22/22	Sat 3/05/22	12	0%	9
2.2	Find Contours	Mon 3/07/22	Fri 3/11/22	5	0%	5
2.3	Overlay Virtual Keyboard Map	Mon 3/14/22	Tue 3/15/22	2	0%	2
2.4	Methodology Writing	Wed 3/16/22	Fri 3/18/22	3	0%	3
3	**Finger Detection and Tracking, Integration**	Mon 3/21/22	Wed 4/13/22	24		18
3.1	Finger Detection and Tracking	Mon 3/21/22	Fri 3/25/22	5	0%	5
3.2	Integration (Find ROI)	Mon 3/28/22	Fri 4/01/22	5	0%	5
3.3	Integration (Find and Return Fingertip on ROI)	Mon 4/04/22	Fri 4/08/22	5	0%	5
3.4	Methodology Writing	Mon 4/11/22	Wed 4/13/22	3	0%	3
4	**Touch Typing Trainer**	Thu 4/14/22	Thu 5/19/22	36		26
4.1	Front-end Development	Thu 4/14/22	Sat 4/23/22	10	0%	7
4.2	Back-end Development	Sun 4/24/22	Tue 5/03/22	10	0%	7
4.3	Finger-key Identification and Mapping Accuracy	Wed 5/04/22	Fri 5/13/22	10	0%	8
4.4	Methodology; Results and Discussion Writing	Sat 5/14/22	Thu 5/19/22	6	0%	4
5	**Final Paper Writing and Presentation**	Fri 5/20/22	Wed 6/15/22	27		19
5.1	Conclusion	Fri 5/20/22	Sun 5/22/22	3	0%	1
5.2	Recommendation and Future Work	Mon 5/23/22	Wed 5/25/22	3	0%	3
5.3	Proofreading	Thu 5/26/22	Fri 6/03/22	9	0%	7
5.4	Journal Article Writing	Sat 6/04/22	Fri 6/10/22	7	0%	5
5.5	Presentation Preparation	Sat 6/11/22	Wed 6/15/22	5	0%	3



References

About TypeRacer. (2021). Retrieved October 26, 2021, from https://data.typeracer.com/misc/about

Alexanderson, S., O'Sullivan, C., & Beskow, J. (2016). Robust online motion capture labeling of finger markers. *Proceedings of the 9th International Conference on Motion in Games*, 7–13. https://doi.org/10.1145/2994258.2994264

ANSI INCITS 154-1988. (1999). *Office Machines and Supplies - Alphanumeric Machines - Keyboard Arrangement* (Standard). American National Standards Institute. Washington, D.C., USA.

Apple. (2021). How to identify your Apple keyboard layout by country or region. Retrieved October 18, 2021, from https://support.apple.com/en-ph/HT201794

Arif, A., & Stuerzlinger, W. (2009). Analysis of text entry performance metrics, 100–105. https://doi.org/10.1109/TIC-STH.2009.5444533

Baker, N., Cham, R., Cidboy, E., Cook, J., & Redfern, M. (2007). Kinematics of the fingers and hands during computer keyboard use. *Clinical Biomechanics*, *22*(1), 34–43. https://doi.org/10.1016/j.clinbiomech.2006.08.008

Baker, N., Cham, R., Hale, E., Cook, J., & Redfern, M. (2007). Digit kinematics during typing with standard and ergonomic keyboard configurations. *International Journal of Industrial Ergonomics*, *37*(4), 345–355. https://doi.org/10.1016/j.ergon.2006.12.004

Bartnik, J. (2021). https://monkeytype.com/about

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Ciobanu, O., Gavat, C., & Cozmei, R. (2016). The keyboard remains the least economically designed computer device. https://doi.org/10.1109/EHB.2015.7391585

Common Core State Standards Initiative (CCSS). (2010). *Common core state standards for English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjucts* (tech. rep.). Retrieved October 26, 2021, from http://www.corestandards.org/wp-content/uploads/ELA_Standards1.pdf

Department of Education. (2020). *Contextualized Mechanisms to guide Schools in the implementation of the Blended/Distance Learning Delivery Modality* (tech. rep.). Retrieved October 26, 2021, from https://www.depedcar.ph/sites/default/files/regionalMemos/long-memo_1.pdf

Dobson, A. (2009). *Touch Typing in Ten Hours*. Hachette UK.

Donica, D. K., Giroux, P., & Faust, A. (2018). Keyboarding instruction: Comparison of techniques for improved keyboarding skills in elementary students. *Journal of Occupational Therapy, Schools, & Early Intervention*, *11*(4), 396–410. https://doi.org/10.1080/19411243.2018.1512067

Dorfmuller-Ulhaas, K., & Schmalstieg, D. (2001). Finger tracking for interaction in augmented environments. *Proceedings IEEE and ACM International Symposium on Augmented Reality*, 55–64. https://doi.org/10.1109/ISAR.2001.970515

Doyle, A. C. (1999). *The Adventures of Sherlock Holmes*. Retrieved February 3, 2022, from https://www.gutenberg.org/ebooks/1661

DraugTheWhopper. (2014). MS Natural Multimedia Keyboard. Retrieved October 28, 2021, from https://commons.wikimedia.org/wiki/File:MS_Natural_Multimedia_Keyboard.png

Ergodox EZ. (n.d.). ErgoDox EZ: An Incredible Mechanical Ergonomic Keyboard. Retrieved October 28, 2021, from https://ergodox-ez.com/

Fisher, B. (n.d.). Bilateral Filtering. Retrieved May 18, 2022, from https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

Gerr, F., Marcus, M., Ensor, C., Kleinbaum, D., Cohen, S., Edwards, A., Gentry, E., Ortiz, D., & Monteilh, C. (2002). A prospective study of computer users: I. Study design and incidence of musculoskeletal symptoms and disorders. *American Journal of Industrial Medicine*, *41*(4), 221–235. https://doi.org/10.1002/ajim.10066

Hoot, J. L. (1986). Keyboarding Instruction in the Early Grades: Must or Mistake? [Publisher: Routledge _eprint: https://doi.org/10.1080/00094056.1986.10521749]. *Childhood Education*, *63*(2), 95–101. https://doi.org/10.1080/00094056.1986.10521749

Hsu, M. H., Shih, T. K., & Chiang, J. S. (2014). Real-Time Finger Tracking for Virtual Instruments. *2014 7th International Conference on Ubi-Media Computing and Workshops*, 133–138. https://doi.org/10.1109/U-MEDIA.2014.53

Huang, T. (1996). Computer Vision : Evolution And Promise [Publisher: CERN]. https://doi.org/10.5170/CERN-1996-008.21

ISO/IEC 9995-1:2009. (2016). *Information technology — Keyboard layouts for text and office systems* (Standard). International Organization for Standardization.

Joyce, J. (2003). *Ulysses*. Retrieved February 3, 2022, from https://www.gutenberg.org/ebooks/4300

Keyboarding Without Tears — K-5. (2020). Retrieved October 26, 2021, from https://issuu.com/handwritingwithouttears/docs/kwtbrochure2020/1

Keybr.com - Typing lessons. (2021). Retrieved October 26, 2021, from http://www.keybr.com/

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector [arXiv: 1512.02325]. *arXiv:1512.02325 [cs]*. https://doi.org/10.1007/978-3-319-46448-0_2

Logan, G., Ulrich, J., & Lindsey, D. (2016). Different (key)strokes for different folks: How standard and nonstandard typists balance Fitts' law and Hick's law. *Journal of Experimental Psychology: Human Perception and Performance*, *42*(12), 2084–2102. https://doi.org/10.1037/xhp0000272

Logitech C920 PRO HD Webcam, 1080p Video with Stereo Audio. (n.d.). Retrieved January 26, 2022, from https://www.logitech.com/en-ph/products/webcams/c920-pro-hd-webcam.html

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (n.d.). Hands. Retrieved January 13, 2022, from https://google.github.io/mediapipe/solutions/hands.html

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A Framework for Perceiving and Processing Reality. *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*. https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf

Marklin, R. W., Simoneau, G. G., & Monroe, J. F. (1999). Wrist and forearm posture from typing on split and vertically inclined computer keyboards. *Human Factors*, *41*(4), 559–569. https://doi.org/10.1518/001872099779656770

MathWorks. (2021a). Computer Vision Toolbox. Retrieved October 28, 2021, from https://www.mathworks.com/products/computer-vision.html

MathWorks. (2021b). What Is MATLAB? Retrieved October 28, 2021, from https://www.mathworks.com/discovery/what-is-matlab.html

Matt3o. (2014). Filcom MINILA Air pictured with Logitech M705 mouse for scale. Retrieved October 28, 2021, from https://deskthority.net/wiki/File:Filco_MINILA_Air.jpg

Melville, H. (2001). *Moby Dick; Or, The Whale*. Retrieved December 7, 2021, from https://www.gutenberg.org/ebooks/2701

Norman, D. A., & Fisher, D. (1982). Why Alphabetic Keyboards Are Not Easy To Use: Keyboard Layout Doesn't Much Matter. *Human Factors*, *24*(5), 509–519. https://doi.org/10.1177/001872088202400502

OpenCV: Color Space Conversions. (n.d.). Retrieved May 18, 2022, from https://docs.opencv.org/4.5.5/d8/d01/group__imgproc__color__conversions.html

OpenCV: Contours : Getting Started. (n.d.). Retrieved December 14, 2021, from https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

OpenCV: Geometric Image Transformations. (n.d.). Retrieved May 18, 2022, from https://docs.opencv.org/4.5.5/da/d54/group__imgproc__transform.html

OpenCV: Image Filtering. (n.d.). Retrieved May 18, 2022, from https://docs.opencv.org/4.5.5/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed

Operations, V. (2021). Does Typing Cause Carpal Tunnel? 3 Myths About Carpal Tunnel Syndrome. Retrieved October 26, 2021, from https://minnesotavalleysurgerycenter.com/pain-management/does-typing-cause-carpal-tunnel-3-myths-about-carpal-tunnel-syndrome/

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms [Conference Name: IEEE Transactions on Systems, Man, and Cybernetics]. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076

Parkkinen, T. (2018). The ultimate guide to keyboard layouts and form factors [Section: Keyboard Information]. Retrieved October 18, 2021, from https://blog.wooting.nl/the-ultimate-guide-to-keyboard-layouts-and-form-factors/

Poole, D., & Preciado, M. (2016). Touch typing instruction: Elementary teachers' beliefs and practices. *Computers and Education*, *102*, 1–14. https://doi.org/10.1016/j.compedu.2016.06.008

Ripat, J., Giesbrecht, E., Quanbury, A., & Kelso, S. (2010). Effectiveness of an ergonomic keyboard for typists with work related upper extremity disorders: A follow-up study. *Work*, *37*(3), 275–283. https://doi.org/10.3233/WOR-2010-1079

Rumudiez. (2013). Correctly labeled modifier keys for the ANSI Keyboard layout. Retrieved October 18, 2021, from https://commons.wikimedia.org/wiki/File:ANSI_Keyboard_Layout_Diagram_with_Form_Factor.svg

Saalfeld, A. (1999). Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm [Publisher: Taylor & Francis _eprint: https://doi.org/10.1559/152304099782424901].

Cartography and Geographic Information Science, *26*(1), 7–18. https://doi.org/10.1559/152304099782424901

Serina, E. R., Tal, R., & Rempel, D. (1999). Wrist and forearm postures and motions during typing. *Ergonomics*, *42*(7), 938–951. https://doi.org/10.1080/001401399185225

Shelley, M. W. (1993). *Frankenstein; Or, The Modern Prometheus.* Retrieved February 3, 2022, from https://www.gutenberg.org/ebooks/84

Sobel, I. (2014). An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*.

Suzuki, S., & Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46. https://doi.org/10.1016/0734-189X(85)90016-7

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions [ISSN: 1063-6919], *07-12-June-2015*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Szeto, G., Straker, L., & O'Sullivan, P. (2005). A comparison of symptomatic and asymptomatic office workers performing monotonous keyboard work - 2: Neck and shoulder kinematics. *Manual Therapy*, *10*(4), 281–291. https://doi.org/10.1016/j.math.2005.01.005

Toosi, K. K., Hogaboom, N. S., Oyster, M. L., & Boninger, M. L. (2015). Computer keyboarding biomechanics and acute changes in median nerve indicative of carpal tunnel syndrome. *Clinical Biomechanics*, *30*(6), 546–550. https://doi.org/10.1016/j.clinbiomech.2015.04.008

UMass Amherst. (n.d.). An Overview of Quizzes in Moodle — UMass Amherst Information Technology — UMass Amherst. Retrieved October 26, 2021, from https://www.umass.edu/it/support/moodle/overview-quizzes-moodle

Wheatland, N., Wang, Y., Song, H., Neff, M., Zordan, V., & Jörg, S. (2015). State of the Art in Hand and Finger Modeling and Animation. *Computer Graphics Forum*, *34*(2), 735–760. https://doi.org/10.1111/cgf.12595

Yousaf, M., & Habib, H. (2014). Virtual Keyboard: Real-Time Finger Joints Tracking for Keystroke Detection and Recognition. *Arabian Journal for Science and Engineering*, *39*(2), 923–934. https://doi.org/10.1007/s13369-013-0909-2