Accessible Two-handed Gesture Control for Human Computer Interaction

Patricia Ouyang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-116
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-116.html

May 13, 2022

Accessible Two-handed Gesture Control for Human Computer Interaction

Patricia Ouyang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Brian Barsky
Research Advisor

13 May 2022

(Date)

* * * * * * *

Professor Eric Paulos
Second Reader

13 May 2022

(Date)

Abstract

Accessible Two-handed Gesture Control for Human Computer Interaction

by

Patricia Ouyang

Master of Science in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Brian Barsky, Chair

The mouse and keyboard were first invented decades ago and still remain the most ubiquitous forms of computer input today along with the trackpad. In order to control a modern computer, many people must operate physical hardware that may pose challenges for certain people and cause discomfort and pain for others if used over long periods of time.

We propose a two-handed gesture control system for people who find it difficult to use a traditional mouse and keyboard. The system uses solely a laptop webcam for input and allows users to perform common laptop actions and tasks through combinations of left and right hand gestures, requiring no additional hardware. Through performing gesture recognition analysis to evaluate our gesture recognition methods and user feedback, we determine that our two-handed gesture control system is a functional and accessible form of alternative control.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to acknowledge and thank Xinying Hu, an M.Eng. student and fellow research collaborator who provided the implementation for the affine transformation method for gesture recognition, fingertip method for volume control, and browser control mode for our project. Special thanks to Professor Brian Barsky for his guidance and valuable insight throughout the research process, and my other project group members: M.Eng. students Fang Hu, Yang Huang, Guanghan Pan, and Juntao Peng. I would also like to thank my family for their continued support.

Chapter 1

Introduction

1.1 Motivation

For people with disabilities, using a traditional mouse, trackpad, or keyboard as a form of input for a laptop or computer may come with difficulties that make use of said technology very challenging or impossible. Even for individuals without disabilities, use of a mouse or keyboard for extended periods of time can cause discomfort and strain that may lead to developing "technological diseases" such as carpal tunnel syndrome [19]. Alternative input devices do exist, but are oftentimes designed for specific impairments and thus unsuitable for certain users. Ergonomic mice and keyboards are available, but can be very costly.

However, as opposed to hardware, assistive technology in the form of software can provide an alternative form of input that is both customizable and adaptable to the user's needs. With this in mind, we developed a camera-based two-handed gesture recognition system that maps different combinations of gestures to different computer controls, allowing for customizable input that only relies on the user's ability to form certain hand gestures.

1.2 Overview

Our project aims to allow users to interact with a laptop solely through their hand gestures and the laptop's webcam. A hand can be modeled as a connected group of key points, with each key point being an important joint or identifiable landmark of the hand. Static hand gestures are gestures that can be classified into a predefined number of gesture categories relying solely on the static image of the hand at a point in time without considering hand motion. In our system, static hand gesture recognition is achieved through relative key point

position or image processing using affine transformations. The recognized hand gestures can then be used as input to the two-handed control scheme.

1.3 Related Work

Mouse and Keyboard Alternatives

There are currently many hardware alternatives available for cursor control that aim to reduce strain caused by using a traditional mouse on the wrist and other parts of the hand, including vertical mice, trackballs, joysticks, etc. Hardware alternatives for traditional keyboards include on-screen keyboards, one-handed keyboards, and expanded keyboards [11]. An example of a more complicated hardware setup includes a head-mounted mouse and finger pressure sensors that rely on head movements, masticatory muscle contractions, and finger contact pressure to control mouse and keyboard input [6]. Many hardware alternatives are designed for people with specific impairments and thus may work well for some, but may not be an ideal form of input for others.

Alternative Control Schemes

In terms of software, different eye tracking systems have been proposed that allow for users to control the cursor through gaze detection [15, 22]. Eye tracking software such as Precision Gaze Mouse [16] and GazePointer [5] are a readily available options for cursor control through eye tracking. Head tracking is another common alternative for cursor movement and functionality. One such example would be using head position to determine cursor position and triggering cursor functions such as clicking through blinking [14]. Systems that incorporate a combination of both eye tracking and facial muscle detection have also been proposed [3].

One drawback of eye and head tracking for cursor control is that some eye and head movements may be involuntary or unintentional and may make cursor control difficult. There are also limitations to the unique number and types of movements that can be made with the eyes and head, which may limit cursor control functionality. The wide variety of possible hand gestures as input may remedy this situation in our system.

Some hand gesture recognition control schemes utilize a combination of software and external hardware. GWindows is an example of using hand gestures and additional video cameras to perform window management tasks. The cameras provide depth information

that is needed for the gesture recognition. Our gesture recognition system does not require depth information, and thus requires no additional hardware besides the laptop's built-in webcam.

Gesture Recognition

Currently, a lot of hand gesture recognition research use deep learning and convolutional neural networks to classify static and dynamic hand gestures [13, 17, 18]. Other non-machine learning classification methods involve additional depth cameras or having the user wear specific hardware to collect additional data [4, 12]. In order for our system to not be restricted to only using hand gestures present in the hand gesture datasets needed to train the machine learning models, we opted to instead use relative key points and image processing techniques that do not require large datasets for effective implementation. Our methods also do not require additional hardware for our system to be usable, making it more lightweight and accessible to different users.

Chapter 2

Approach

2.1 System Design and Control Pipeline

Our two-handed gesture recognition control system takes in solely video data from a laptop's webcam as input in order to detect the user's two hands frame-by-frame. When detected, each hand is overlayed with a skeleton showing the hand's key points. Hand detection and hand key points are determined through Google's MediaPipe framework [20].

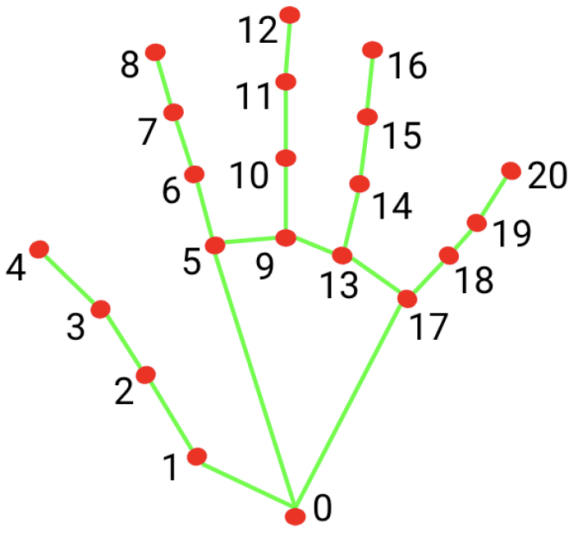


Figure 2.1: MediaPipe hand model key points

Hand key points for the left and right hand are continuously determined for each frame of video input. The key points are then input into our static hand gesture recognition

system to determine the current hand gestures of the user. Two different approaches are used for hand gesture recognition: a heuristic using relative key point position, and an image processing technique using affine transformations between images. Once the hand gestures are classified, the gestures are used as input to the two-handed control scheme in order to to control the laptop and determine what action the user is trying to accomplish, such as moving the cursor, scrolling, typing, etc.

Hand gestures are mapped to different laptop controls, with the left hand controlling the general mode of control and the right hand providing finer control options for the mode specified by the left hand, or the specific action. Actions are performed only through a combination of left and right hand gestures and movement. If the user changes their left or right hand gesture, it will be detected by the system and the input to the control scheme will change. This will cause the old action to stop and the new, intended action to be carried out (if it exists). Below is a figure summarizing the general control system.

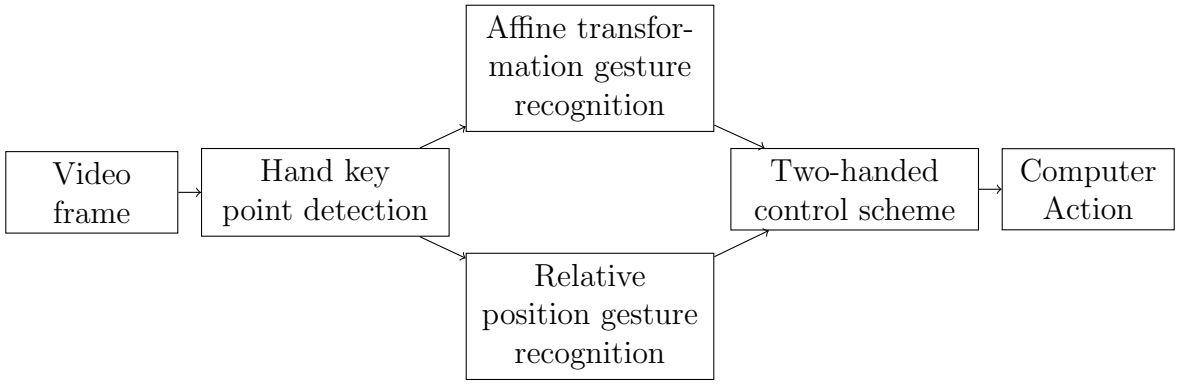


Figure 2.2: Two-handed gesture control system

Customizability and Past Approaches

Due to the wide range of hand gestures possible and the added flexibility of two hands, the number of unique gesture combinations are theoretically infinite. The gestures used in our mapping scheme should ideally be intuitive in terms of what action the gesture is carrying out as well. For these reasons, we want our system to be able to easily accommodate new gestures that could potentially be added when expanding the system's functionality or improving the system's ease of use.

Machine Learning-based Gesture Recognition

We previously explored using various machine learning algorithms [1, 7, 8] in order to carry out hand gesture recognition, but found that this method relied heavily on training models using preexisting hand gesture datasets such as [2, 23], which would not be ideal for our goals. Although it may be possible to find a large hand gesture dataset containing many different gestures to train on such that each gesture can be classified by the model, adding a new gesture to our gesture mapping control scheme would create complications if the gesture does not exist in the current dataset. This meant that our system would be confined to only using the set of hand gestures available in the single dataset used to train the model if we were to use this approach, which would limit the system's functionality and customizability.

Two-handed Dynamic Gestures

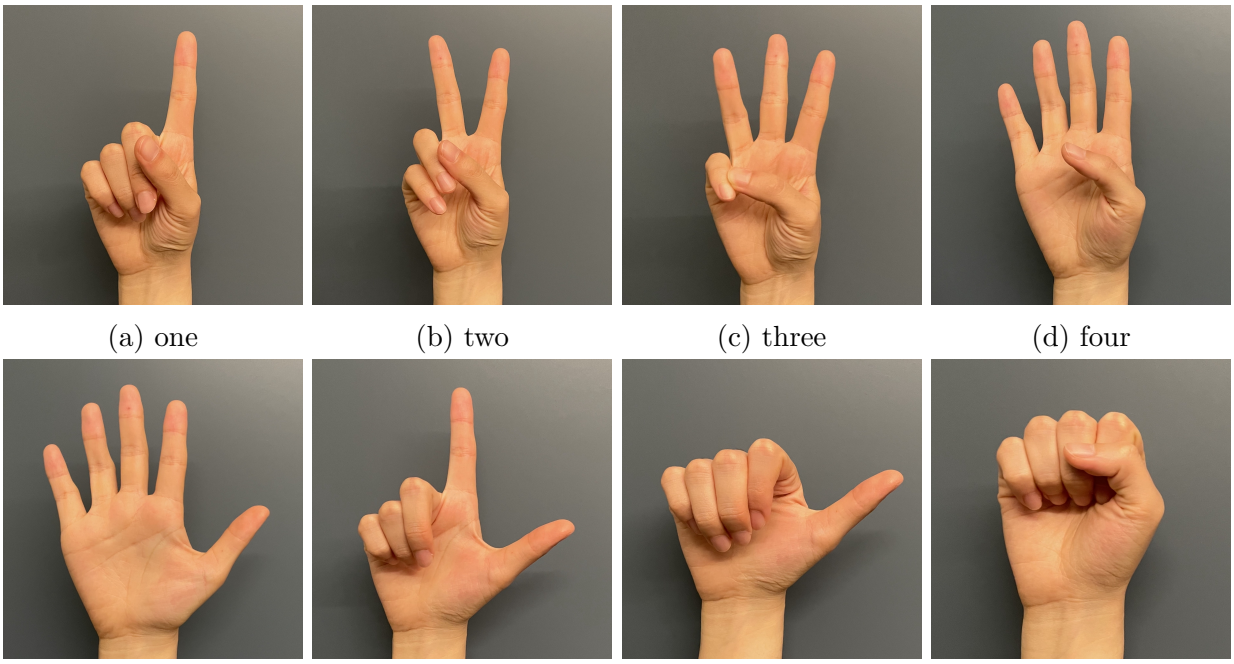
Unlike static hand gestures, dynamic hand gestures incorporate hand movement in order to convey different gestures. Two-handed, dynamic hand gestures involve both hands moving at the same time to perform a single gesture, which may be a more intuitive form of control for the user as opposed to both hands operating at the same time, but performing different gestures separately. Previous works have explored the possibility of manipulating objects on a screen with two-handed dynamic gestures using a video camera [10]. The Xbox One, developed by Microsoft, has the option for two-handed dynamic gesture controls–however, the controls require use of external hardware, the Kinect [24], for motion and depth-sensing. In fact, many current two-handed dynamic gesture recognition methods require a depth camera, making them unsuitable for our accessibility-oriented, webcam-only control scheme requirements.

2.2 Gesture Recognition

Our system is able to classify a total of 8 static hand gestures: numbers 1 through 5, arrow, thumb, and fist.

Relative Key Point Position Method

The Mediapipe framework provides the hand key point data needed for this method, specifically the x and y coordinates of 21 key points for a each hand, as well as whether or not the hand detected is the left or right hand (handedness). We assign a state to each finger



(a) one (b) two (c) three (d) four

(e) five (f) arrow (g) thumb (h) fist

Figure 2.3: Gestures supported by our system.

and determine each finger's current state using the key points. The states are initialized as 'UNKNOWN', and can alternate between 'OPEN' and 'CLOSE' as determined by the relative position of the key points. For all fingers except the thumb, 'OPEN' corresponds to the finger pointing upwards and 'CLOSE' corresponds to the finger folded downwards towards the palm. The state is determined by the relative y coordinates of the key points for each finger. For the thumb, 'OPEN' corresponds to the thumb facing outward and 'CLOSE' corresponds to the thumb tucked inwards towards the palm. The thumb state depends on a combination of the handedness and the relative x and y coordinates of the thumb key points. The current gesture is then determined using the combination of the finger states. See Figure 2.4 for an example.

Affine Transformation Method

An affine transformation is a function that can perform both translation and linear transformations. In the 2D case, given a minimum of 3 points that do not lie on a line and their corresponding transformed outputs, we can determine a unique affine transformation matrix that can perform the same transformation on future input points.

Figure 2.4: The thumb state is 'CLOSE' and all other finger states are 'OPEN' for the 'four' gesture shown when using relative position gesture recognition.

The affine transformation method is inspired by [21] and also detailed in [9]. For each hand gesture, we save an image of said gesture which serves as a "template" or basis of comparison for future gestures that we are trying to classify. The key points for each template image are also determined. See Figure 2.5 for an example of template key points. Additional gesture templates and template key points can be generated to add more gestures to our system using this method.

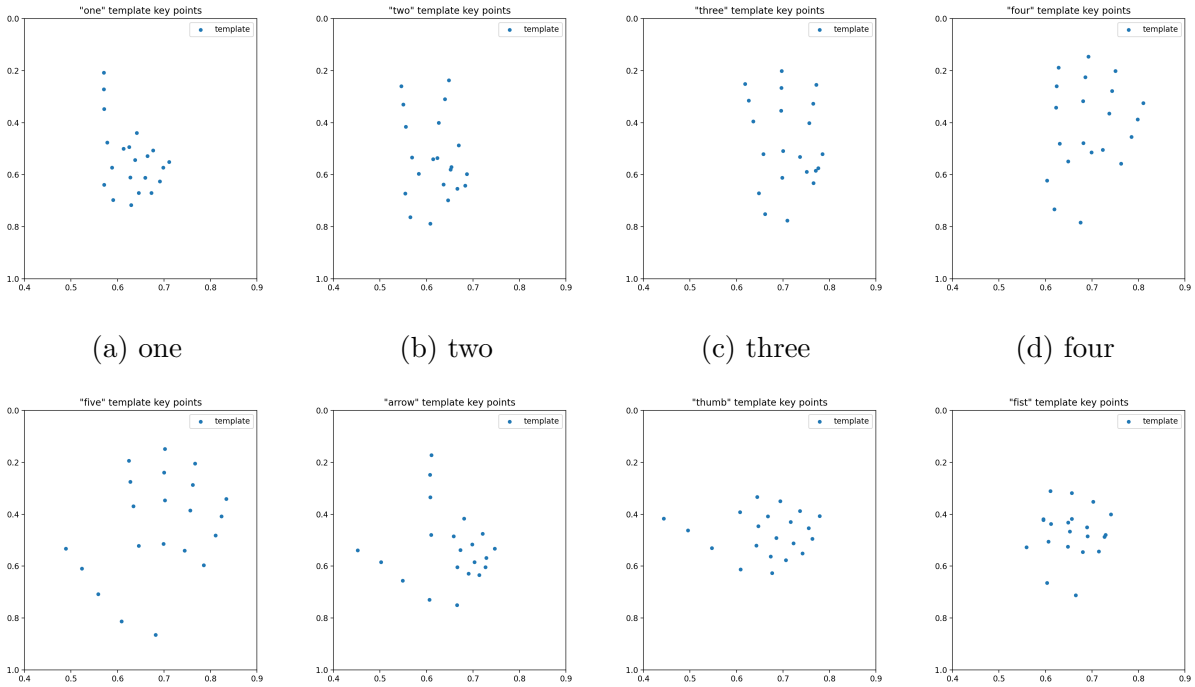
When classifying a single hand in a given frame, the key points from the hand in the current frame F and the corresponding key points in each template image T are used in order to compute 8 different optimal affine transformation matrices. Computing the affine transformation matrix M for a given template reduces to the following problem:

$$M^* = \arg\max_M ||M \cdot F - T||$$

Each affine transformation matrix is then applied to the original key points to obtain 8 sets of transformed key points. Ideally, the transformed points would be closer to the template key points for the correct gesture and farther away for incorrect gestures. Therefore, the transformed points are then compared against each template's key points to determine the error between them, which is then converted into a score using the following equation:

$$score = exp(-||H^* \cdot F - T||)$$

The gesture with the highest score is chosen as the final prediction. See Figure 2.6 for a visual comparison of a set of transformed input gesture key points with multiple templates.



(a) one (b) two (c) three (d) four

(e) five (f) arrow (g) thumb (h) fist

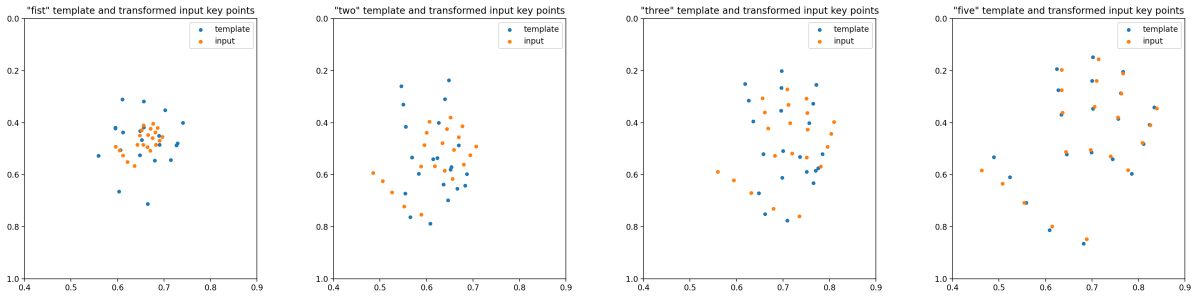
Figure 2.5: Affine transformation gesture recognition template image key points.

2.3 Two-handed Gesture Mapping

The two-handed control system maps the left hand gesture to different modes and the right hand gesture to specific actions given the mode determined by the left hand. The mode mappings for the left hand are as follows: 1 - cursor, 2 - scroll, 3 - volume, 4 - window, 5 - browser, arrow - keyboard. Mode mappings can be switched and additional mappings can be added in order to accommodate more control options and extend system functionality.

Cursor Control

In cursor mode, the right hand controls the cursor movement and actions such as clicking, double-clicking, and right-clicking. Different cursor actions can be mapped to any gesture the system is able to recognize. Cursor movement is based on the absolute position of the hand in the frame mapped proportionally to the screen. For the purposes of cursor control, the position of the hand is reduced to the center of the palm, which is determined by the key



(a) fist template (b) two template (c) three template (d) five template

Figure 2.6: Affine transformation gesture recognition template images with transformed 'five' input gesture key points.

point coordinates. The cursor follows the hand's movement across the screen as the hand moves across the frame. Other cursor control methods such as relative control and joystick control were implemented, but absolute cursor control was found to be the most intuitive for practical use.

Relative Cursor Control

Relative cursor control imitates the behavior of a real mouse or trackpad–moving a mouse quickly in a certain direction or moving a finger quickly across a trackpad should result in the cursor moving a greater distance across the screen as opposed to moving slowly. This allows for the entire screen to be accessible with little actual physical movement, and the user would not have to reach across the entire frame, as is the case with absolute cursor control. For relative cursor control, temporal information between frames is used to determine the change in hand position and the hand speed over the past few frames. The speed of the cursor increases as the speed of the hand increases. Adjustments can also be made to customize how much faster the cursor moves depending on the hand speed. Since small hand movements can be adjusted to hardly move the cursor, this method is robust to noise such as hand tremors.

Relative cursor control is not as intuitive in practice since there is no physical hardware the user can touch in order to gauge how fast their hand should move. This approach is also very sensitive to lag–small delays between hand movement and cursor movement on the screen makes gauging the appropriate hand speed rather difficult.

Joystick Cursor Control

Joystick cursor control is modeled after video game joystick controls. A point is set as the center of control, and cursor movement is based on the hand's position relative to said center point. For example, if the user's hand is to the left of the point, the cursor will move continuously to the left on the screen. The speed at which the cursor moves depends on the hand's distance from the set point; greater distance corresponds to faster cursor movement, similar to a physical joystick. The cursor will remain still when the hand is at the center point. Since it is almost impossible to have the hand remain at the exact center point when the user does not wish to move the cursor, a radius around the center point can be set such that any hand movement within the specified circle determined by the radius and center point will not result in any cursor movement.

This method of control is also not very intuitive to users since it is possible to continue moving the cursor without any actual hand movement. This problem is exacerbated by the lack of hardware that a user can touch when using a real joystick that allows the center point to become more tangible. The center point can be hard to keep track of as a circle in a frame that the user must internalize when holding their hand up in front of the camera.

Absolute Cursor Control

Absolute cursor control, described above, is the most intuitive cursor control method for users since the cursor movement reflects hand movement exactly in terms of both both speed and direction. However, despite being intuitive, this method has drawbacks as well. For example, with an exact mapping of the video frame to the entire screen, the user must move their right hand across the frame and reach all the way to the very left of the frame if they wish to move their cursor to the corresponding left edge of the screen. When the hand nears the edge of the frame, it is also harder to perform gesture recognition since some parts of the hand may get cut off while the palm center remains inside the frame. Forcing the hand to move unnaturally outside of a comfortable range also causes the hand contort and angle itself differently, making gesture recognition less accurate compared to when the hand is directly facing the camera. Hand gestures are also harder for the user to perform comfortably when the hand is contorted in such a way.

To remedy these problems, we set a bounding box smaller than the frame and proportionally map the bounding box area to the same screen area such that only hand movement within the bounding box moves the cursor. This allows the hand to move a lot less in order to cover more distance, makes the screen edges reachable without cutting off parts of the

hand, and allows the hand to move within a comfortable range and perform gestures with ease. The bounding box position and dimensions can also be adjusted by the user.

Additionally, since the webcam resolution is oftentimes lower than the screen resolution of a laptop, small changes in hand position will cause the cursor to move many pixels across the screen. This makes the absolute cursor control method very sensitive to noise such as small unintentional movements or hand shaking. The sensitivity is further heightened by the bounding box as the box area decreases. Users must make rather precise movements while also trying to remain within the bounding box, which may be even more difficult if the target area is small, such as a small icon. However, while this method may not be the most robust, it is the most intuitive method for cursor control without a physical component that users can touch.

Scrolling and Volume Control

Scrolling

In scroll mode, the right hand controls the direction of the scrolling (up, down, left, or right). If the right hand is not detected to be performing one of the four gestures mapped to each direction, no scrolling occurs. Users can adjust the speed for continuous scrolling. One limitation is that the scroll speed, while adjustable, is set to that speed for all scrolling; using the control scheme, users only control the scroll direction and not the speed. This means that users cannot scroll slowly and then scroll quickly as they please in the same session without stopping to adjust the scroll speed. One way to control the scroll speed and direction would be to use an approach similar to the joystick cursor control method; the right hand's position relative to some center point is used to determine the scroll direction, and the hand's distance from the center point is used to determine the scroll speed.

Volume Control

In volume control mode, the right hand controls increasing, decreasing, muting, and unmuting the sound output using three different gestures. Muting and unmuting is mapped to the same gesture that acts as a toggle. Volume control using gesture mapping is also sensitive to lag–if the volume is not adjusted within a timeframe that the user expects, they may accidentally hold the gesture for too long and cause the volume to become very loud or very soft. Since each frame of recognized gesture input would cause the volume to change in this

mode, in order to make sure that the volume does not increase or decrease too quickly, we make sure that a certain amount of time has passed between volume increments.

An alternative approach we explored is to have the right hand control the volume through the distance between the index finger and thumb tips instead of through gesture mapping; if the tips of the two fingers are touching then the volume would be 0, effectively muting the sound, and the volume increases as the distance between the fingertips increases. This approach mitigates the effects of lag since the user would just need to hold their hand in the same position until they achieve a certain volume and adjust accordingly. Using the gesture mapping method, users tend to overshoot how long to hold a certain gesture when lag occurs, and then overcompensate when trying to readjust in the opposite direction due to the volume being too high or too low, making it harder to adjust to the ideal volume.

The fingertip distance method is based on the perceived 2D distance between the fingertips, meaning that if the user pushes their hand towards the screen or draws their hand away from the screen without changing the fingertip distance, the volume also changes unintentionally. Additional depth information of the hand key points would be needed in order to mitigate this problem. However, since there is usually not a significant change in hand distance from the screen when a user is performing gestures, this issue does not greatly impact the effectiveness of the finger distance method for volume control. Thus, volume control using the fingertip distance method seems to be a suitable alternative to gesture mapping.

Window Management and Browser Control

Window management and browser control actions are not continuous and do not rely on a steady stream of input for control and adjustment, in contrast to previously described actions such as cursor movement, scrolling, and volume control. Non-continuous mouse actions such as clicking are relatively harmless if done twice in succession against the user's will, but window management and browser control actions may cause great inconvenience if repeated unintentionally due to the right hand performing the same gesture for too long. To safeguard against this, we introduce a slight delay of a few frames between actions in this mode such that two actions will not performed too quickly one after another, similar to incrementing the volume using gesture mapping control. This gives the user enough time to recollect their right hand.

Window Management

In window management mode, the right hand's gestures are mapped to basic window management functions such as closing and minimizing the current window, as well as basic application management such as restoring the previous application and being able to view all currently running applications in order to select one.

Browser Control

In browser control mode, the right hand's gestures are mapped to common browser and tab management actions such as creating a new tab, closing a tab, switching tabs, zooming in or out, and jumping to the address bar to type. Browser actions are executed by mapping gestures to different keyboard shortcuts, so any action that can be performed through a keyboard shortcut can be added to the control scheme as needed by the user. Although keyboard shortcuts are browser-dependent, many common shortcuts are universal across most browsers.

Keyboard Control

Unlike the other modes, the left hand gesture that maps to keyboard control mode is a toggle that turns the mode on or off when detected. Keyboard mode is intended primarily for typing purposes. When keyboard mode is toggled on and currently active, the unique combinations of left and right hand gestures map to the 26 English letters and 10 Arabic numerals, as well as the spacebar, backspace, and return keys. When toggled off, the control scheme reverts back to the original scheme with the left hand specifying the mode of control. Since typing consists of non-continuous keyboard actions, keyboard mode also makes use of an enforced frame delay between actions to ensure users input one character at a time instead of a string of multiple of the same character when typing.

For the letters of the alphabet, due to the large number of letters, the combinations of left and right hand gestures must be easy to remember for the user, usually by following some sort of intuitive order that corresponds to the order of the alphabet. Our system maps the left hand to groups of letters, with individual letters within a group specified by the right hand.

The alphabet is separated into five groups of letters in alphabetical order, each containing five letters except for the last group which contains six. The first group contains a, b, c, d, and e, the second group contains f, g, h, i, and j, etc. The last group contains u, v, w,

x, y, and z. The first through fifth groups are mapped to left hand gestures 1 through 5 respectively. Within a group, the letters are mapped to gestures 1 through 5 for the right hand in alphabetical order. For example, to type the letter h, the user would use a gesture combination 2 and 3 for the left and right hand respectively to specify the third letter in the second group. The letter z is arbitrarily mapped to some other right hand gesture that is not 1 through 5 with the left hand gesture being 5. Below is a table summarizing the typing scheme for the letters of the alphabet.

		Right hand					
		1	2	3	4	5	?
Left hand	1	a	b	c	d	e	n/a
	2	f	g	h	i	j	n/a
	3	k	l	m	n	o	n/a
	4	p	q	r	s	t	n/a
	5	u	v	w	x	y	z

Figure 2.7: Typing control scheme

To avoid users having to calculate what gestures to use for a certain letter, the letters in a group are displayed to the user through the webcam video output when the user's left hand maps to a certain group. The user can then cycle through multiple groups quickly to find the letter needed. A similar scheme is used for the digits 0 through 9.

Although keyboard mode provides a functional method for typing, it currently does not map to all of the keys that a user has access to when using a standard keyboard. A more complex mapping system would be required if all keys were to be mapped. Additionally, the current scheme requires the user to either remember all of the mappings or look at the webcam video to ensure that they are performing the correct gestures, which further reduces typing speed.

For clarity, the two-handed gesture control scheme is summarized in the figure below.

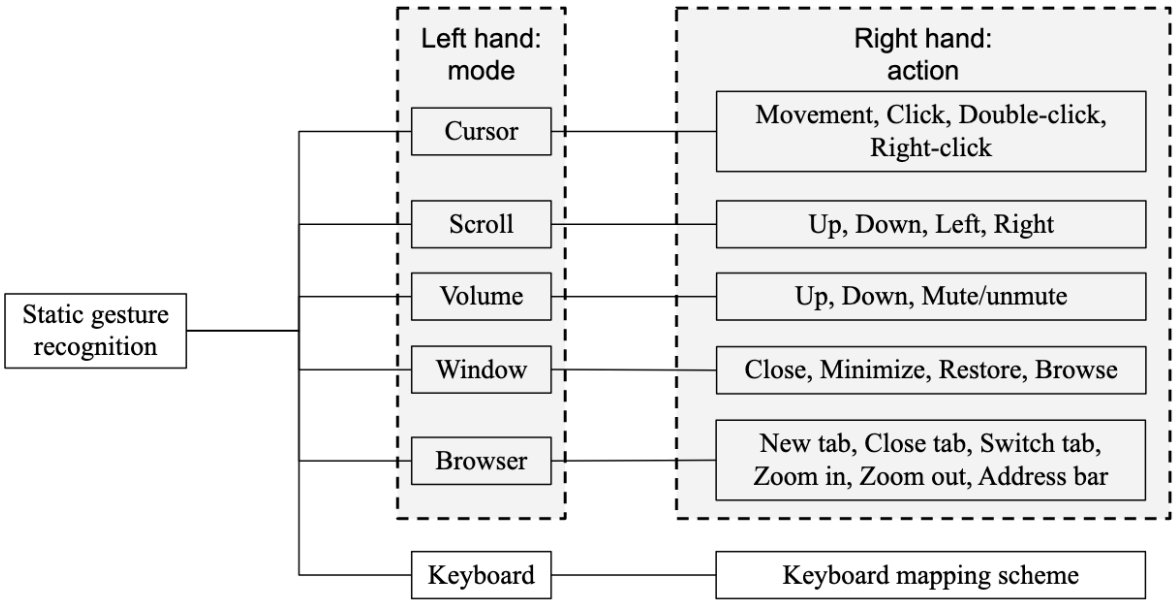


Figure 2.8: Two-handed gesture control scheme

Chapter 3

Results

3.1 Gesture Recognition Analysis

We evaluated the accuracy of the two methods for gesture recognition used in our system: the relative position method and the affine transformation method. The accuracy was determined by the percentage of gestures that the two methods were able to classify correctly based on our tests.

We generated a total of ten sets of frames for both models to classify. To generate a set of frames, we ran our system once and saved each video frame as an image. While the system was running, we cycled through the eight unique static hand gestures recognized by our system using the right hand in quick succession before stopping the system. A set of images was saved, along with the corresponding key points of the hand in each frame and the output of the relative position gesture recognition method. We then used the key points from each image to determine the output of the affine transformation gesture recognition method.

We repeated this process nine more times and labeled the frames with the true hand gesture. Images that were too blurry or ambiguous such as frames that capture the hand transitioning between gestures were removed. Half of the sets were generated with the right hand in a normal position directly facing the camera, while the other half were generated with the right hand incorporating slight deviations such as not directly facing the camera or being slightly farther away or closer to the screen in order to determine how robust the two methods are to slight hand positioning and gesture irregularities that the system would be likely to encounter.

In the first five sets, the hand gestures were performed directly facing the camera. In

Frame Set	Accuracy (%)	
	Rel.	Aff.
1	100.0	96.8
2	86.0	100.0
3	98.3	100.0
4	91.7	100.0
5	100.0	98.3

Table 3.1: Gesture Recognition Method Accuracies.

terms of individual gesture accuracies, the relative position method had the most trouble classifying the 'fist' gesture correctly, while the affine transformation method tended to misclassify the 'thumb out' gesture. The affine transformation method performed better than the relative position method on average. This may be due to the fact that the relative position method is not as forgiving when it comes to key points being out of place; if even one of the relative position requirements is violated for a certain gesture classification, said gesture is immediately ruled out. The affine transformation method, on the other hand, will always pick the gesture with the lowest error or distance between the imput and the gesture template key points, making it the more forgiving method since the classification is based on the key points as a whole instead of looking at individual points.

In the last five sets, the hand gestures were performed with a slight hand variation for each set–further away from the screen, tilted backward, tilted forward, tilted towards the right, and closer towards the screen respectively. Since the relative position method is not as robust to hand variations, the accuracy was lower on average compared to its performance on the first five sets, which aligned with our expectations. The affine transformation method still performed better on average, although the average accuracy also decreased compared to its average from the first five sets. Both methods were more robust to the 2D transformations (scaling and rotation) compared to the backward and forward tilts. Overall, the affine transformation method seemed to be both more accurate and robust compared to the relative position method.

It is worth noting that our data collection was not without bias. The number of frames that were labeled as each certain gesture were not the same due to the way we collected the frames, so some gestures were weighted more heavily when computing the accuracies. Frame labeling and removing a frame due to it being too blurry or ambiguous was also up to our interpretation, although we tried to remain impartial and consistent. Finally, the frames

Frame Set	Accuracy (%)	
	Rel.	Aff.
6	93.4	100.0
7	89.5	87.7
8	82.4	89.7
9	100.0	100.0
10	83.8	98.5

Table 3.2: Gesture Recognition Method Accuracies Cont.

were collected by someone who is familiar with the system, which may have impacted the way the gestures were performed. We expect the accuracies to be lower if the gestures were performed by someone who had never used the system before.

3.2 User Feedback

We conducted an informal user survey in order to gather feedback on our system's design, usability, and practicality in different settings. Previously, the majority of the system's development and refinement was based on our own testing and usage environment, which is not very diverse and also not representative of our target audience. By obtaining user feedback from a wider array of individuals, we can better understand potential complications that may be encountered when using traditional laptop control methods, as well as the shortcomings of our own system, which we can take into account when improving our system to better suite the needs of all kinds of users.

In general, users that participated in the survey found that they were able to adjust basic laptop settings and perform most actions successfully using solely the two-handed control scheme. However, adjustments and actions that required more precision were not always easy to carry out, such as adjusting the volume to a very specific level or trying to click a particularly small icon. Very fine cursor control using absolute cursor control was also more sensitive to hand tremors.

Users also found it difficult to remember the gesture mappings for all of the modes and actions, and had to constantly look at the webcam video (where the current mode and possible actions are displayed) for reference, although the more common actions were soon memorized after a period of familiarization with the system. Some actions were also mapped to gestures that users found to be not very intuitive.

When using the system for long periods of time, users found that the control scheme was easier to use as a whole, but more difficult to use when trying to carry out actions for an application that the system does not have a specific mode for due to the lack of custom controls for said application. However, most users were able to eventually accomplish their intended actions. Overall, users understood the advantages of the two-handed gesture control system after using the system.

Chapter 4

Discussion

4.1 Limitations and Improvements

Although our two-handed gesture control system provides users with the means to control their laptop for basic workflows, the system is still not able to completely replace the functionality of the traditional mouse or trackpad and keyboard combination in certain aspects. The current system also has certain limitations that can be improved.

First, the system is currently reliant on including text on the webcam video in order to display important information such as the currently recognized gestures for the left and right hand, the current mode, and the available actions. The webcam video text is also used for showing additional details such as letter groupings for keyboard input and the bounding box when constraining the proportionally mapped area for absolute cursor control. Our system would benefit from a better user interface that would not require the user to have to constantly switch between their current task and the webcam video application.

Since some users may find certain hand gestures more intuitive or easier to carry out for certain actions compared to others, being able to easily add recognizable gestures and remap hand gestures to different actions would greatly improve the user experience. Our system is currently capable of adding new gestures and remapping gestures, but the user is not able to do this themselves. The improved user interface could include an option for users to start a flow that walks them through adding a new gesture, as well as an interface for remapping gestures and customizing gesture combinations.

The current keyboard input scheme is also slower compared to typing using a keyboard. Linking keyboard input to on-screen keyboards or investigating alternative forms of input for text such as speech or sign language may provide more insight to improving typing speed.

The keyboard input functionality can also be improved to allow for pressing and holding a key and pressing two keys at the same time at the will of the user, which would be useful for keyboard shortcuts and other applications such as gaming.

Finally, our system is currently only available for the Mac OS, although it is possible to extend it to other operating systems.

4.2 Future Work

There are still many different approaches and topics that can be explored in the assistive technology and human-computer interaction space. Future research directions for our project include but are not limited to the following.

Dynamic Gesture Recognition

Our system is currently only able to recognize and utilize static hand gestures. Incorporating dynamic hand gestures provides users with even more options for control due to the addition of hand motion when performing gestures. Dynamic gestures can then also be mapped to different actions. Some actions may also be more intuitive with dynamic gestures, such as using an up and down swiping motion to scroll instead of holding the hand in a static position.

Depth Information

Our system currently uses x and y coordinates of the hand key points in a frame without considering the third dimension or z coordinate, which would encode how far away the hand is from the screen. Depth information would be useful for dynamic gesture recognition and may also improve currently existing functionality such as making gesture recognition more robust to certain hand tilts and the issue of volume changing due to the hand moving closer towards or farther away from the screen when using the finger tip distance method for volume control. MediaPipe already includes the z coordinate when calculating key points, which can potentially be utilized if depth information is to be included in our system. If we no longer wish to constrain ourselves to only using a webcam, another option would be to use additional hardware such as a depth camera–the cost of adding hardware may be offset by the advantages that the additional depth information provides.

Hand Tremor Filtering

Our system uses absolute cursor control for cursor movement, which is more sensitive to noise and hand tremors compared to other cursor control methods. Hand tremor filtering can be incorporated to improve the quality of absolute cursor control such that the cursor is not as sensitive to slight changes in hand position. Given that the cursor's position is based on the palm center, one straightforward approach is to only move the cursor if the palm center deviates more than a certain distance from an average of the palm centers in the past couple of frames. More sophisticated hand stabilization techniques may yield better results.

Speech Recognition and Sign Language

Speech recognition and sign language are alternate forms of input that do not require a mouse and keyboard and thus may be useful additions to our system. Actions can be triggered by the user saying or signing a certain word or phrase, such as "click" for clicking. Speech recognition and sign language would also be applicable to our system's typing scheme as a way for users to easily to convey words that are to be translated into text input.

Chapter 5

Conclusion

Our two-handed gesture control system provides an accessible and functional alternative to the traditional mouse and keyboard. By utilizing only a webcam and hand gestures, users are able to control their laptop and perform common actions and tasks. In particular, people who find using traditional methods of control challenging may benefit from our system. As technology becomes more widespread and its set of users more diverse, we believe that our two-handed gesture control system is a step towards making technology usage more inclusive for everybody.

Bibliography

[1] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[2] Congqi Cao, Yifan Zhang, Yi Wu, Hanqing Lu, and Jian Cheng. "Egocentric Gesture Recognition Using Recurrent 3D Convolutional Neural Networks with Spatiotemporal Transformer Modules". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3783–3791. DOI: 10.1109/ICCV.2017.406.

[3] Craig A. Chin, Armando B. Barreto, J. Gualberto Cremades, and Malek Adjouadi. "Integrated electromyogram and eye-gaze tracking cursor control system for computer users with motor disabilities." In: *Journal of rehabilitation research and development* 45 1 (2008), pp. 161–74.

[4] Ilya Chugunov and Avideh Zakhor. "Duodepth: Static Gesture Recognition Via Dual Depth Sensors". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2019. DOI: 10.1109/icip.2019.8803665. URL: https://doi.org/10.1109%5C%2Ficip.2019.8803665.

[5] *GazePointer*. https://gazerecorder.com/gazepointer/. Accessed: 2022-04-29.

[6] Diyar Gür, Niklas Schäfer, Mario Kupnik, and Philipp Beckerle. "A Human–Computer Interface Replacing Mouse and Keyboard for Individuals with Limited Upper Limb Mobility". In: *Multimodal Technologies and Interaction* 4.4 (2020). ISSN: 2414-4088. DOI: 10.3390/mti4040084. URL: https://www.mdpi.com/2414-4088/4/4/84.

[7] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. "Support vector machines". In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.

[8] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Fang Hu, Xinying Hu, Yang Huang, Guanghan Pan, and Juntao Peng. *SimpleGest: Assistive Hand-Gesture Recognition Technology Based on Computer Vision*. M.Eng. Capstone Report, University of California, Berkeley. May 2022.

[10] Agnès Just and Sébastien Marcel. "Two-Handed Gesture Recognition". In: (Jan. 2005).

[11] *Keyboard and mouse alternatives and adaptations*. https://abilitynet.org.uk/factsheets/keyboard-and-mouse-alternatives-and-adaptations. Accessed: 2022-04-29.

[12] Matej Králik and Marek Šuppa. *WaveGlove: Transformer-based hand gesture recognition using multiple inertial sensors*. 2021. DOI: 10.48550/ARXIV.2105.01753. URL: https://arxiv.org/abs/2105.01753.

[13] Hasan Mahmud, Mashrur M. Morshed, and Md. Kamrul Hasan. *A Deep Learning-based Multimodal Depth-Aware Dynamic Hand Gesture Recognition System*. 2021. DOI: 10.48550/ARXIV.2107.02543. URL: https://arxiv.org/abs/2107.02543.

[14] Eric Missimer and Margrit Betke. "Blink and Wink Detection for Mouse Pointer Control". In: *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*. PETRA '10. Samos, Greece: Association for Computing Machinery, 2010. ISBN: 9781450300711. DOI: 10.1145/1839294.1839322. URL: https://doi.org/10.1145/1839294.1839322.

[15] Mohamed Nasor, Mujeeb Rahman K k, Maryam Mohamed, Haya Ansari, and Farida Mohamed. "Eye-controlled mouse cursor for physically disabled individual". In: Feb. 2018, pp. 1–4. DOI: 10.1109/ICASET.2018.8376907.

[16] *Precision Gaze Mouse*. https://precisiongazemouse.org/. Accessed: 2022-04-29.

[17] Abir Sen, Tapas Kumar Mishra, and Ratnakar Dash. *A Novel Hand Gesture Detection and Recognition system based on ensemble-based Convolutional Neural Network*. 2022. DOI: 10.48550/ARXIV.2202.12519. URL: https://arxiv.org/abs/2202.12519.

[18] Jing-Hao Sun, Ting-Ting Ji, Shu-Bin Zhang, Jia-Kui Yang, and Guang-Rong Ji. "Research on the Hand Gesture Recognition Based on Deep Learning". In: *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*. 2018, pp. 1–4. DOI: 10.1109/ISAPE.2018.8634348.

[19] Merita Tiric Campara, Ferid Krupic, Mirza Biscevic, Emina Spahic Dervisevic, Kerima Maglajlija, Zlatan Masic, Lejla Zunic, and Izet Masic. "Technological Diseases: Carpal Tunnel Syndrome, a Mouse Shoulder, Cervical Pain Syndrome". In: *Acta Informatica Medica* 22 (Oct. 2014), pp. 333–340. DOI: 10.5455/aim.2014.22.333-340.

[20] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. *MediaPipe Hands: On-device Real-time Hand Tracking*. 2020. DOI: 10.48550/ARXIV.2006.10214. URL: https://arxiv.org/abs/2006.10214.

[21] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul L. Rosin, Zixi Cai, Han Xi, Dingcheng Yang, Hao-Zhi Huang, and Shi-Min Hu. "Pose2Seg: Detection Free Human Instance Segmentation". In: (2018). DOI: 10.48550/ARXIV.1803.10683. URL: https://arxiv.org/abs/1803.10683.

[22] Xuebai Zhang, Xiaolong Liu, Shyan-Ming Yuan, and Shu-Fan Lin. "Eye Tracking Based Control System for Natural Human-Computer Interaction". In: *Computational Intelligence and Neuroscience* 2017 (Dec. 2017), pp. 1–9. DOI: 10.1155/2017/5739301.

[23] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. "EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition". In: *IEEE Transactions on Multimedia* 20.5 (2018), pp. 1038–1050. DOI: 10.1109/TMM.2018.2808769.

[24] Zhengyou Zhang. "Microsoft Kinect Sensor and Its Effect". In: *IEEE MultiMedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/MMUL.2012.24.