Designing an Assistive Mouse for Human Computer Interaction Using Hand Gestures

Michael Qi

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-119
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-119.html

May 14, 2021

Acknowledgement

I would like to thank Professor Brian Barsky for his guidance throughout this project. I would also like to thank the team members of my research group: Master of Engineering students Frank Cai, Sihao Chen, Xuantong Liu, Weili Liu, Yizhou Wang, and Shiqi Wu as well as undergraduates Yash Baldawa, Raghav Gupta, Rohan Hajela, Varun Murthy, Viansa Schmulbach, Mengti Sun, and Sirinda Wongpanich.

Designing an Assistive Mouse for Human Computer Interaction Using Hand Gestures

Michael Qi

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 2021

(Date)

* * * * * * *

Professor Eric Paulos
Second Reader

14 May 2021

(Date)

Abstract

While computers have become increasingly prevalent in recent years, they are not accessible to all people. Although technology has advanced tremendously, human computer interaction systems have not evolved to the same degree since their conception. In fact, the traditional computer mouse used today was first designed more than 50 years ago. However, as computer users grow increasingly diverse, the limitations of the mouse become more apparent. To provide a greater degree of flexibility for a wide variety of people, we must develop alternative systems of human computer interaction.

We propose an assistive mouse for people who are unable to use a traditional mouse comfortably due to physical limitations. It uses hand tracking and gesture recognition to enable cursor movement and mouse actions, respectively. Furthermore, it incorporates anti-shake filters to help people with essential tremors. Through our evaluation, we have confirmed that each of the component modules achieves high accuracy and precision. As a result, the system is currently operable.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

For people with disabilities, interacting with computers using conventional methods can be a challenge. In 2012, an estimated 10 million people suffered from essential tremors in the United States alone [9]. Similarly, certain individuals lack the fingers required to comfortably operate a physical mouse. Alongside its potential for causing hand strain [17], the shortcomings of the traditional mouse have become increasingly evident. While it is the most popular method of computer control by far, it is by no means perfect.

Unfortunately, there are currently very few alternatives to the traditional mouse. Although several unorthodox devices do exist for people with disabilities [33] and ergonomic options are offered, many of them are quite expensive. Furthermore, recent years have seen a shift away from hardware solutions in favor of software ones. Our research group developed a camera-based assistive mouse controlled by hand movements and gestures as an alternative method of human computer interaction.

1.2 Background

In our system, we aim to emulate the two main components of a mouse. The first, cursor movement, can be achieved through hand tracking. Meanwhile, mouse actions such as clicking and dragging will be performed through gesture recognition.

Hand tracking identifies the pixels corresponding to hands in a picture. Under specified circumstances, researchers have seen success using algorithms such as background subtraction

[30, 38] and color segmentation [10, 35] to solve this problem. However, with the recent advent of deep learning, neural networks have surged in popularity as a more powerful and flexible approach to hand tracking. Given enough labelled data, convolutional neural networks can detect hands more accurately and reliably than image processing techniques [32]. Also, many neural networks also deviate from prior image processing approaches to hand tracking by identifying only the locations of specified hand key points [21, 39]. These models summarize hands as skeletons of their most important joints to greatly accelerate labelling and training time.

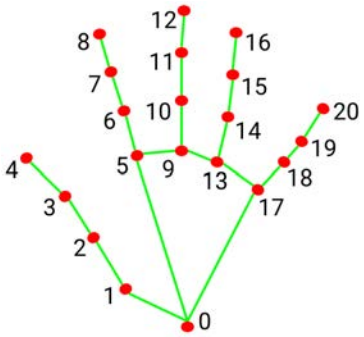

Figure 1.1: Hand key point layout of Google's MediaPipe model

Meanwhile, the goal of gesture recognition is to classify the hand action performed in an image. Image processing approaches accomplish this by using the specific shape of fingers [7] or identifying convexity defects [8, 37]. However, these methods tend to only work with a small number of static gestures. In contrast, machine learning algorithms operating on video input can support dynamic gestures while also offering greater classification accuracy. Some models, known as single shot detectors, report only the predicted gesture from the input [18]. However, more efficient systems first track the hand, then apply a classifier to a subset of the image which is determined by the bounding box of the hand [22, 31]. Finally, using a smaller input of key points instead of the entire image can greatly improve the performance speed of gesture recognition.

1.3 Related Work

Currently, the most prevalent solutions for assistive mouse software are built on head tracking. For example, both Camera Mouse [3] and Apple [20] have developed systems to control the cursor using head movements. However, head tracking does not support a wide range of clicking functionality. While facial expression analysis has been explored to mitigate this [13],

it is limited to a small number of classes. In our proposed system, we hope to offer greater flexibility in movement and features by recognizing hand motion and gestures, respectively.

In addition, due to the computational intensity required for many machine learning methods, researchers often choose to instead use image processing techniques for hand tracking and gesture recognition to perform at a sufficient number of frames per second (FPS) [37, 38]. However, optimizations in neural network architecture have allowed recent models to run quickly without the need for graphics processing unit (GPU) hardware. We explore both approaches to create a robust assistive mouse for human computer interaction that operates in real time.

A proof of concept for the assistive mouse has already been demonstrated [2]. It provides a overview of hand tracking and gesture recognition by examining the advantages and limitations of various approaches. To build on this foundation, we incorporate more complex algorithms and recent machine learning developments into our system. In this work, we also focus on the transition from these modules to human computer interaction to provide an intuitive assistive mouse solution.

Chapter 2

Methods

2.1 Image Processing Algorithms

Failed Approaches

Contour-Based Solutions

An ideal hand tracking system can detect hands in an image without the need for calibration. In Canny edge detection [5], this can be achieved by locating areas of high contrast. Through the use of image gradients and hysteresis thresholding, this algorithm identifies the most prominent lines and contours in a picture. Depending on their shape, specific objects can then be recognized and segmented.

Unfortunately, this method alone is insufficient for locating hands in a complex background. Given an image, Canny edge detection only outputs the pixels with high local contrast, and it does not provide any insight on their context. As a result, there is no guarantee that each contour only belongs to a single object. However, pixel variations, and by extension, edges, are often created by the intersection of multiple objects. Determining which portions of the contours correlate to hands is a challenging problem that cannot be solved by Canny edge detection.

Furthermore, in images with many objects, the hand rarely corresponds to a single connected contour. Instead, it is often comprised of many smaller line segments. Despite our attempts to consolidate these pieces into a cohesive outline, we were unable to find a method to reliably perform this task. Morphological operations [12] such as dilation can close small gaps, but they also connect the hand with unrelated nearby contours. Similarly, clustering

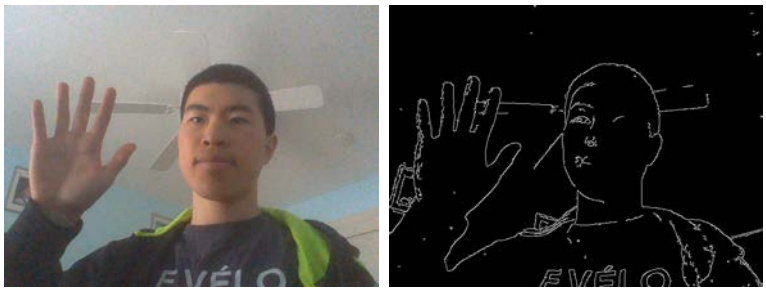

Figure 2.1: Canny edge detection

algorithms exhibit the same limitations when grouping edges. While their parameters can be adjusted to successfully segment the hand on individual images, this approach is not robust to even small variations. Finally, decreasing hysteresis parameters or using adaptive thresholding can detect weaker edges. Even so, this also creates additional noise. As a result, we ultimately concluded edge detection was not a reliable enough approach to hand tracking.

CamShift

CamShift [1] is an alternative method for hand tracking that utilizes the temporal nature of video input. After defining a bounding box in an initial image, the algorithm locates the region in subsequent frames with the most similar pixel distribution. In other words, it can follow any object it is initialized to. Furthermore, it will even adjust the size of its bounding box to better fit the current image.

While inconvenient, the calibration required in CamShift is not a major drawback. Manual initialization of the hand is possible, but this can be challenging for users of an assistive mouse. Since the operation only needs to be performed once, a better solution is to use a stronger machine learning classifier for the initial frame. Even weaker hardware will not experience a noticeable decline in performance with a single hand detection, and CamShift can proceed smoothly afterwards.

However, the CamShift algorithm also has multiple limitations. To begin with, its dependence on raw pixel colors makes it difficult to ensure robustness. For example, in images where the background is colored similarly to the hand, CamShift will frequently lose track of the hand and instead focus on a stationary point. In addition, although CamShift supports changing object sizes, it is less flexible to changes in shape. Users perform a variety of gestures to trigger actions in our assistive mouse, making CamShift unsuitable for this purpose. Lastly, CamShift is unable to consider negative samples. For our assistive mouse, recognizing the absence of hands is as important as detecting their presence. Otherwise, erratic behavior

will occur, rendering the system disruptive when hands are not in the frame. This problem could be solved defining a similarity threshold for pixel distributions, but it does not address CamShift's other drawbacks.

Proposed Method

Background Subtraction

Background subtraction is a straightforward yet effective algorithm for hand tracking. First, the algorithm captures an initial frame of the background. All subsequent frames are subtracted from this image, and by calculating the difference between the two, contrasting pixels can easily be identified [30, 38]. As a result, background subtraction can quickly recognize hand movements from when it is initialized.

Due to small variations in lighting conditions and camera error, the result of background subtraction will often include minor noise. To address this issue, we apply a threshold on the intensity of the pixel differences so that only major changes are detected.

Another limitation of background subtraction is that it only functions when the user's background remains static. In other words, it will fail if the user moves their camera or changes their lighting conditions. However, we concluded users of the assistive mouse were unlikely to adjust their background frequently, if at all. Therefore, background subtraction would still serve as an effective approach to hand tracking.

Figure 2.2: Background, captured frame, and background subtraction

Color Segmentation

Color segmentation [10, 35] is a another useful technique to track hands. Given an image, it preserves only the pixels within a specified range of colors. In other words, minimum and maximum color values can be defined to detect skin, and consequently, hands, in a picture.

In practice, research [19] has shown that the HSV color space is better suited than the RGB color space for this purpose.

However, skin color can vary widely, and selecting a narrow range of values will cause skin detection to fail for certain people. On the other hand, a greater number of unwanted background objects will be retained with a wider color range. To address this, we introduced a convolutional neural network to locate the hand when it first enters the image. Our system then sets its thresholds by sampling the detected hand, allowing the approach to support all skin colors. If lighting conditions change or a different user operates the assistive mouse, the color segmentation module can be quickly and easily recalibrated.

Finally, we combined background subtraction and color segmentation to create a more effective hand tracking system. Background subtraction eliminates background elements that fall within the defined color range. Meanwhile, color segmentation removes factors such as clothes. The resulting image will only contain pixels corresponding to the user's skin, including hands.

Figure 2.3: Background subtraction, image mask, and color segmentation

Haar Cascade Classifiers

Haar cascade classifiers are lightweight machine learning models that can be used to detect patterns in images [34]. Unlike neural networks that learn features in hidden layers, cascade classifiers operate using a set of predefined Haar features. Given a large amount of training data, the algorithm can determine whether a specific pattern exists in a picture. Face detection, in particular, has emerged as a prominent application of Haar classifiers, which are able to learn the similarities of human facial features.

In our hand tracking system, the only objects left in the image after color segmentation are the hand and face. Using Haar face detection, the two can be differentiated, and the face can be removed from the image. After applying a dilation filter to close small gaps [29],

only one large shape remains. By extracting this contour, we can successfully segment the hand from the original image.

Figure 2.4: Face removal, dilation, and hand segmentation

However, since Haar cascade classifiers are designed to identify patterns, a more straightforward approach would be to recognize hands directly. Unfortunately, this does not work well in practice. While faces generally maintain the same shape, hands exhibit more variation while performing different gestures. Therefore, a single Haar classifier would be insufficient for hand tracking. Instead, every hand position would require its own model, and the computational cost of applying each one to an image would invalidate the performance advantages of the classifier. In addition, Haar classifiers are not especially robust. Although it is not evident in face detection, the algorithm does not support rotations. If the head is turned or tilted around even 30 degrees, a Haar classifier will fail to locate it. This is a minor issue for face detection, but much more problematic for hand tracking. One objective of the assistive mouse is to increase flexibility, and this is not possible with a Haar cascade classifier for hands. Even so, face detection can still be used to segment hands.

Palm Center Location

With the hand successfully segmented, our assistive mouse can use its position for cursor control. However, the hand is comprised of numerous pixels, and they must be reduced to a single coordinate. We chose to use the palm center for this purpose. The fingertips and mean pixel will frequently change by performing different gestures even if the hand position has not changed, making precision controls difficult. Meanwhile, the palm center remains relatively constant no matter what hand gesture is being performed [37].

A common approach to locate the palm center and radius is to calculate the maximum inscribed circle of the hand [23]. By finding the largest circle contained with the contour, the palm can be approximated while ignoring the fingers. In continuous space, exact solutions to this problem achieve $O(n \log n)$ complexity with respect to the number of points on the

contour. As the size of the hand increases, so does the time required to determine the maximum inscribed circle.

Fortunately, in discrete pixel space, a more straightforward method exists to locate the palm center. The distance transform calculates the space from each white pixel to the nearest black pixel. Therefore, by applying it to a filled hand contour, we can identify the palm center and radius in $\theta(n)$ time [7]. If the contour were not filled, gaps in the segmented hand would lead to incorrect results. However, filling the hand also inadvertently closes space in gestures such as hand circles. While inconvenient, this is a trivial drawback compared to the robustness provided by the fill operation.

Finally, in the case that the user is wearing short sleeves, there are situations where the maximum inscribed circle is located on the forearm near the elbow. To prevent this, we limit our search to the top of the hand. More specifically, pixels below 150% of the hand's width are not considered when calculating the maximum inscribed circle.

Figure 2.5: Segmented hand, distance transformation, and maximum inscribed circle

Static Gesture Recognition

The segmented hand is also used to perform static gesture recognition. By analyzing the hand's shape, we can determine the number of fingers shown in an image. Sharp angles in the hand contour often correspond to the space between fingers, and these points can be located using convexity defects [8]. First, a convex hull is constructed containing all the pixels of the segmented hand. Next, any large deviations from this polygon are classified as the space between fingers. Since there is always one more finger than defect, we can use this method to effectively count the number of fingers shown. In the case when no defects are identified, we conclude the user is making a fist.

However, this approach is unable to distinguish between one and two fingers because both gestures consist of a single convexity defect. To differentiate the two, we compare their areas relative to the palm radius. A gesture with one finger will create a much larger convexity

defect than a gesture with two fingers. Therefore, we can set a threshold such that the system correctly classifies these gestures.

There are several limitations to our gesture recognition module. For example, it only identifies the number of fingers shown, not which ones are raised. Therefore, only six gestures are available, but additional ones could be defined by using more complex geometry. Also, our image processing algorithms cannot recognize dynamic gestures. While this provides less flexibility, we believe it is actually beneficial to the assistive mouse as a whole. If dynamic gestures were supported, they would conflict with the hand tracking module. In other words, it would be difficult for our system to distinguish between a hand movement and a swipe gesture. As a result, we decided not to pursue dynamic gesture recognition for the assistive mouse.

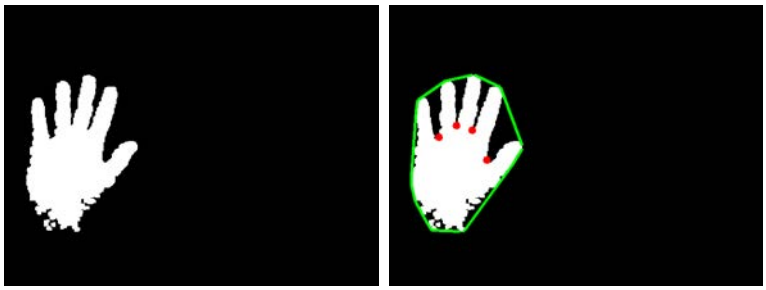

Figure 2.6: Segmented hand and convexity defects

For reference, our proposed method in its entirety is as follows.

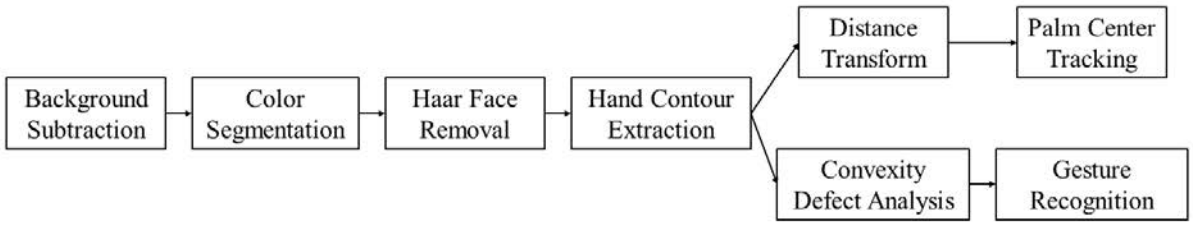

Figure 2.7: Complete image processing algorithm

2.2 Deep Learning Algorithms

Key Point Detection

Deep learning is a more powerful and robust approach to hand tracking than image processing algorithms [24, 28]. Previously, neural networks required GPU hardware to support

the computational intensity of their models. However, recent optimizations have enabled detectors to operate in real time using only CPU resources.

Instead of locating all the pixels corresponding to the hand, these models only identify the most important hand points [6, 39]. Furthermore, the predicted coordinates also include an estimate of depth. Although some image information is lost during inference, an additional axis is added. By learning fewer parameters, these neural networks can converge more quickly than convolutional models. As a result, they are affected less by picture noise.

In fact, the hand skeleton is even more useful than the hand's pixels for palm tracking. Key points are more stable than the distance transform, which is sensitive to segmentation noise. Although none of the key points correspond directly to the palm center, it can be calculated using nearby key points. To do so, we construct a polygon from the points surrounding the palm. Using Gauss's area formula, we can determine its centroid. With the key point detector, palm tracking is easily achievable.

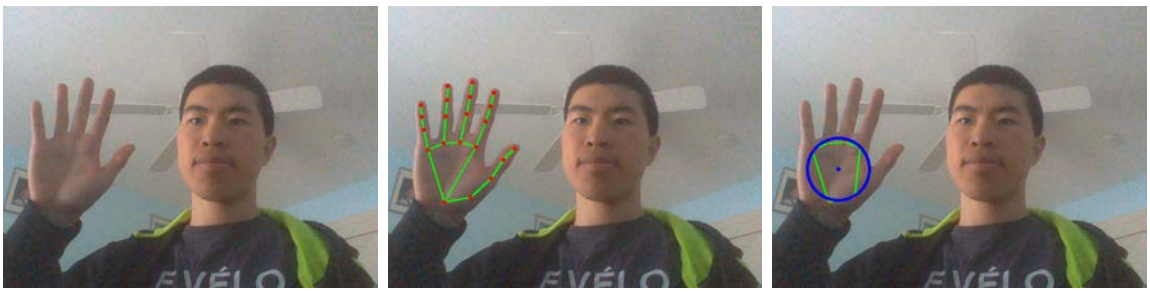

Figure 2.8: Captured frame, key points, and palm centroid

Key Point Gesture Recognition

Key points can also be used to perform gesture recognition. While it is possible to use a geometric analysis similar to convexity defects for this purpose, machine learning methods are far more robust. In addition, key points are a much smaller input than the segmented hand. Therefore, training machine learning models can be completed more quickly and with less data. Finally, unlike the segmented hand, key points can be easily preprocessed. For each sample, we rotate the hand to an upright position and normalized its size before using it as input to our algorithms. Wu et al. compared the classification accuracies of several machine learning algorithms to find the best model for gesture recognition [36].

Support Vector Machines

A support vector machine (SVM) finds boundaries to separate data in higher dimensional space [14]. In our assistive mouse, the optimal hyperplanes partition the samples from each gesture into different areas of the space. However, since the data are not linearly separable, SVMs are unable to achieve perfect accuracy. To address this issue, we use a radial basis function kernel. This adds dimensionality to the training data, allowing the model to learn some nonlinearities.

Decision Trees and Random Forests

Instead of treating each data point as a single input vector, decision trees perform classifications by evaluating one feature at a time [25]. Depending on their value relative to a threshold, different gestures will be predicted, and decision trees can learn nonlinearities quickly using this approach.

Random forests build on decision trees by also incorporating ensemble learning. Increasing the number of classifiers reduces the variance of predictions, which in turn leads to greater accuracy [4]. However, if given the same data, all the decision trees will be identical. To create heterogeneity among the classifiers, each tree is only given a random subset of the features. As a result, their accuracies will vary, and stronger classifiers will be given greater weight in the final prediction.

Neural Networks

Neural networks have become a popular approach to many classification problems such as gesture recognition because of their ability to fit models to high dimensional data [27]. Using hidden weights, this method performs gradient descent to minimize the cost function of its parameters on the training data. Since neural networks are able to model nonlinearities well, we also included the angles between hand key points as input. Next, we used a multilayer perceptron (MLP) network [26] to calculate the probability of each gesture given the key points and angles of each frame. Finally, the gesture associated with the highest value was selected as the prediction. While neural network training can be slow, our models converged quickly due to the small number of features in the data.

However, this approach to gesture recognition classifies each frame independently. Since users are unlikely to change gestures quickly, a better model would leverage the temporal nature of video to make its predictions. A long short-term memory (LSTM) network [15]

uses feedback connections to maintain a hidden state in its weights. In other words, key points in previous frames will impact the current prediction, creating a more accurate and robust gesture recognition model. However, out training data did not fit this architecture as well as the other models. If key point detection failed in a single image, we could remove the sample from the training data, which did not affect classifiers that considered each input independently. For the LSTM network, these samples were retained in the training data. Since the model was trained on consecutive sequences of frames, these points could not be discarded.

2.3 Assistive Mouse Design

Cursor Control Methods

Absolute Cursor Control

In absolute cursor control, the palm's position in the frame is mapped proportionally to the screen. It provides an intuitive method for the user to control the mouse because the cursor always moves in the same direction as the hand.

However, there are several limitations to absolute cursor control. Since web camera image resolution is often lower than the screen resolution, this approach is sensitive to noise. Even small hand movements or shaking will cause the cursor to move across many pixels. As a result, maintaining stability can be difficult for users, especially with those who suffer from conditions such as essential tremors.

Furthermore, mapping the entirety of the frame to the entirety of the screen is not ideal. When controlling the assistive mouse, most users will align their body in the center of the frame. Without a loss of generality, this means a right-handed user must reach across their body to move the cursor to the left side of screen. This problem makes the assistive mouse less accessible to some groups and generates unnecessary inconvenience for users who are able to perform this action. Finally, it is difficult to move the cursor to certain areas of the screen with absolute hand tracking. For example, users will move their hand to the upper corner of the camera frame to close a window. However, if the user's palm is at the corner of the image, their fingers are not visible. Therefore, hand detection will fail, making it impossible to move the cursor to the edge of the screen.

To solve both these problems, we limit absolute cursor control to a smaller area of the image. A window with size equal to one third the image width is visualized on the frame

to represent the boundaries of the cursor control. If the hand is detected outside this area, it is moved to the corresponding edge of the screen. Depending on their preferences or handedness, users can adjust the position of this window.

While this solution does make the entire screen accessible, limiting absolute cursor control to an even smaller region of the frame further increases the sensitivity of the approach. Users found it difficult to keep the cursor in one spot and tended to overshoot precise movements. As a result, we concluded the method was not an effective approach to cursor control.

Relative Cursor Control

Relative cursor control uses the temporal relationship between video frames to move the mouse. Instead of treating each input independently, it moves the cursor based on the change in hand position over the past few frames. As the speed of the hand motion increases, so does the speed of cursor.

This was implemented by retaining the positions of previous palm centers. For each new input coordinate, we computed its difference with the average of the queue. Afterwards, the cursor would move in this direction with distance proportional to the magnitude of the vector.

However, relative cursor control is ineffective when there is a linear relationship between the difference vector and cursor movement. For example, moving the hand five pixels over five frames will result in the same cursor movement as moving the hand five pixels in one frame. A more intuitive system will move the cursor a greater distance as the hand speed increases. To achieve this, for a change Δx, the cursor is moved horizontally $c\Delta x^2$ pixels where c is a scalar constant. By establishing a quadratic relationship between hand movement and cursor control, users can manipulate the mouse more flexibly. Furthermore, the sensitivity c can be adjusted to fit the user's habits. With $c = 1$, the entire screen can be accessed without reaching across the user's body while still offering precision controls. Since small hand movements barely move the cursor, this approach is also robust to minor noise. Similarly, a threshold can be set to ignore small changes in position.

Joystick Cursor Control

Joystick cursor control is inspired by video game systems. The user defines a point on the image as the center, and subsequent cursor movement is informed by the palm's position relative to this point. For example, if the hand is located to the left of the center, the cursor

will move left until the hand is moved. In addition, the cursor's speed is controlled by the distance from the hand to the center. The greater this separation, the quicker the cursor movement.

Out of our three cursor control methods, the joystick is the least sensitive to variation. In absolute control, detection error directly determines the cursor's position, but noise only affects the direction vector in joystick control. In addition, these small pixel variations are generally insignificant relative to the magnitude of this vector. Therefore, even with noise, the cursor will continue to move in the intended direction. This allows the joystick method to function as an extremely robust method of cursor control.

However, the joystick is also the least intuitive of our cursor control systems. Unlike the other approaches, cursor movement can be achieved by keeping the hand still. Similarly, cursor movement does not always correspond to the same direction as hand movement. If the user's hand moves right, the cursor will move left as long as the palm is left of the center. Even so, users were able to familiarize themselves with joystick control with practice.

In addition, maintaining stability can be a challenge with joystick cursor control. Since the direction vector will only be zero if the palm is exactly at the center, there is often slight cursor drift. To solve this problem, we set a threshold for the magnitude of the vector. Any points within this radius are ignored, creating a small dead zone around the center. As a result, the user can place their palm within this circle to keep the cursor stationary.

Tremor and Anti-Shake Filtering

To make our assistive mouse accessible to people with essential tremors, we implemented multiple anti-shake filters. Even if the user attempts to keep their hand steady, noise is unavoidable in hand tracking. Therefore, all users can benefit from filtering.

Simple Moving Average Filter

A simple moving average filter stores the palm centers of the previous frames. It acts as a sliding window, and upon receiving a new coordinate, the mean position is recalculated. This approach blurs the frequency of the hand tracking signal, resulting in smoother motion. In fact, even absolute cursor control is relatively stable after filtering. Depending on the number of frames used to calculate the mean position, control can be tuned to fit the user's preferences. Averaging over a larger number of frames improves stability at the cost of cursor speed, and the opposite holds for a smaller window.

Kalman Filter

The Kalman filter [16] is a more complex approach that uses prior knowledge of the system to predict and filter. Unlike the simple moving average filter, which only stores previous positions, the Kalman filter also tracks velocity. As a result, it is more resistant to changes in direction. The filter also assumes its input will have errors, and both process and measurement uncertainty are considered when receiving a new input. However, tuning these parameters can be a challenge. It requires a good understanding of the uncertainty, but this will vary among users and computers. Even so, some users found the Kalman filter more effective than the simple moving average.

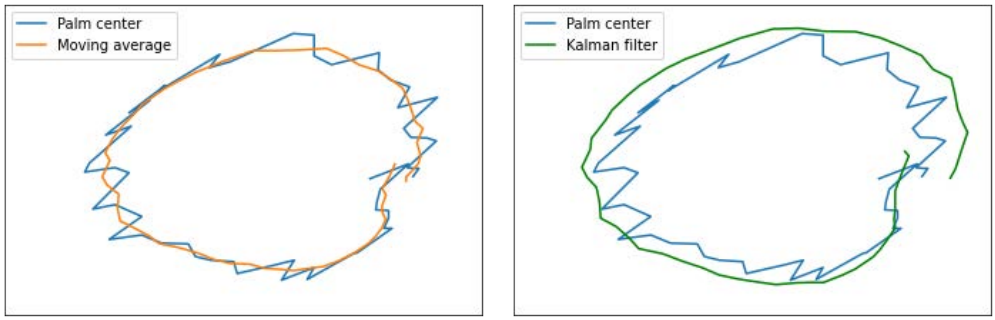

Figure 2.9: Moving average and Kalman filters

Mouse Actions

Mouse functionality can be achieved using our gesture classification models. All of the gestures are mapped to a different mouse action, such as clicking or dragging. When these gestures are recognized, their corresponding action is performed. However, although gesture recognition runs in every frame, the commands should not be executed at the same frequency. For example, the assistive mouse should not continuously click if the correct gesture is shown. Therefore, the system only performs an action when it is first recognized. Users can also remap gestures to different commands according to their preferences and abilities.

Since our gesture recognition models do not achieve perfect accuracy, there will be prediction errors. For mouse actions, recognizing the wrong gesture for even one frame can be extremely disruptive. To solve this issue, similar to cursor control, we also store previous classifications to improve robustness. However, since we cannot calculate a gesture average, we instead use a finite state machine. When the system recognizes a new gesture, it is put in an intermediate state. The corresponding action is performed only after it has been pre-

dicted in a sufficient number of frames. Furthermore, we modified this requirement for each gesture depending on how frequently it is used. While this approach adds a slight delay to mouse functionality, the difference is imperceptible. The system operates with high FPS, and users are unable to distinguish between the finite state machine and immediate mouse actions. Meanwhile, the better stability greatly enhances user experience.

Lastly, we chose palm tracking over alternatives such as fingertip tracking because of its consistency when performing gestures. However, this was not true in practice, and most gestures move the palm center slightly. For example, changing from an open palm to a closed fist lowered the hand position using both image processing methods and key point detection.

In absolute cursor control, this is especially problematic. Users found it challenging to execute the desired action at the correct location using certain gestures. To successfully do so, they were required to offset their hand before making the gesture, which is both unintuitive and inconvenient. For relative cursor control, we solve this problem by clearing the queue of previous palm centers. This way, the cursor will remain stationary when the user performs gestures and respond with the expected behavior. However, this is not an issue for joystick cursor control. Most mouse actions are performed when the cursor is still, and this approach utilizes a dead zone. If it is large enough, users can position their hand such that even if the palm center changes, there will be no cursor movement. Therefore, stable mouse functionality can easily be achieved with the joystick.

2.4 User Research

Lastly, we conducted a small, informal survey to gather feedback on the design of our assistive mouse. Previously, our choices had been informed by personal experience, but they were not representative of our target audience. By receiving input from potential users, we could better comprehend the complications involved in their situations and develop our system to more accurately meet their needs.

As a whole, interviewed participants found our assistive mouse easy to understand. Most of them currently used voice assistants for human computer interaction, and they recognized circumstances where hand tracking would be more advantageous. In addition, they also emphasized the importance of creating an intuitive gesture recognition system. A common suggestion was to assign gestures to logically related functionality, such as holding the thumb down to close a window. Although we address this by allowing users to assign gestures to mouse actions based on their preferences, our system is limited to a set of predefined gestures.

Chapter 3

Results

3.1 Hand Tracking Analysis

Evaluation Methods

We evaluated the accuracy of both the image processing and deep learning hand tracking methods on the stereo hand tracking dataset provided by [40]. This dataset offers hand counting poses across six scenes with different backgrounds. For each image, we calculated the average distance between our identified palm center and the given label in millimeters.

Method	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	All
Image Processing	10.99	14.54	309.88	11.89	13.43	76.99	72.95
Key Point Detection	9.52	11.89	11.98	11.31	12.26	11.61	11.43

To illustrate the distribution of hand tracking error, we also determined the percentage of hand tracking samples below various millimeter thresholds.

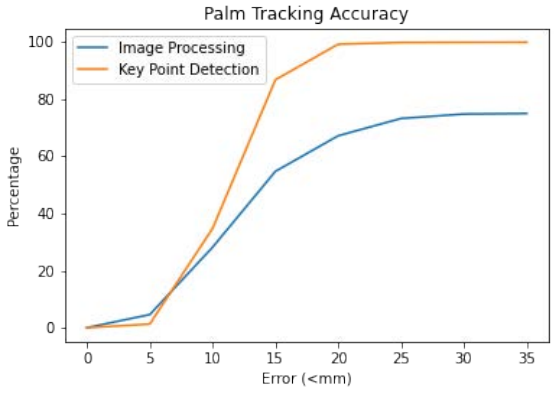

Figure 3.1: Percentage of samples below error

In most scenes, our image processing methods provide accuracy comparable to that of key point detection. However, they fail to correctly locate the hand in two scenes. Key point detection is able to identify the palm center in these backgrounds, and it outperforms our image processing algorithms in all six scenes.

We also performed a performance analysis on both approaches to hand tracking unconstrained by the I/O time required to read images.

Method	Frames per Second
Image Processing	26.44
Key Point Detection	47.36

Key point detection also outperforms our image processing methods in its runtime. In theory, the computation saved in hand tracking can be dedicated to a more complex gesture recognition system. However, the average web camera only captures 30 frames per second, limiting the performance of key point detection. As a result, the difference in processing time will be negligible for most users. Even so, the superior accuracy of key point detection make it the better choice for hand tracking.

3.2 Gesture Recognition Analysis

Evaluation Methods

We trained and evaluated the accuracy of our gesture recognition models on the EgoGesture [41] dataset. To replicate a mouse's functionality, we only required 11 of the 83 available gestures. While dynamic gestures can provide greater intuition for certain commands, we chose to instead use static gestures. Their greater simplicity allows models to classify them more quickly and accurately.

Method	Classification Accuracy
Support Vector Machine	86.25%
Random Forest	92.36%
Multilayer Perceptron Network	88.54%
Long Short-Term Memory Network	82.88%

Support vector machines create the simplest boundaries and have the most difficulty modeling nonlinearities. Although they did not perform as well as our best models, they

still achieved reasonably high accuracy. Our results suggest that hyperplanes alone are insufficient to address the complexity present in hand gestures.

Random forests generated the highest classification accuracy out of all our models. The key points detected for a single gesture frequently exhibit the same spatial patterns, making decision trees and random forests a strong approach to gesture recognition. While slightly less accurate, MLP networks were able to learn these patterns and converge to accurate weights with only a few hidden layers and epochs of training.

Unlike the other models, LSTM networks predict gestures using multiple frames. Although this can lead to greater accuracy, it is also more difficult to train and converges mores slowly. Furthermore, it is more dependent on the accuracy of key point detection, making its predictions less robust to outliers. In simpler models, hand tracking errors will only affect a single frame. Similarly, no classification will occur if key point detection fails. However, in LSTM networks, each frame is used for several predictions. Errors will persist for longer, and the system must compensate when no key points are identified. As a result, training a robust LSTM network requires much greater complexity, leading to its low accuracy in gesture recognition.

3.3 Limitations

As mentioned above, image processing is not as robust as deep learning for tracking hands. Multiple components in our proposed image processing method require initialization to function properly. If the setting is changed even slightly, recalibration may be necessary, and this can lead to a poor user experience. Similarly, our proposed method does not support use of the assistive mouse in a dynamic background although this is not a major inconvenience for most users.

Another limitation of image processing algorithms is that they can lead to unpredictable behavior when the face is occluded by the hand. Since we use the Haar cascade classifier to remove the face from the image, some of the hand will also be removed in certain situations. If the two barely overlap, only parts of the fingers will be lost. While this does not impact hand tracking, it can affect our gesture recognition module. However, if the there is substantial facial occlusion, hand tracking also becomes unstable. When face detection succeeds, its pixels are removed from the image. Therefore, the hand is also lost, and neither palm tracking nor gesture recognition can be executed. Meanwhile, if the face is not detected, the resulting hand contour is misshapen. As a result, palm tracking will fail, and gesture

recognition will be unable to identify fingers. While some work has been done to mitigate this, these approaches are not very robust [11]. For example, they depend on keeping the user's face and body still, but this can be difficult with the assistive mouse. If their hand is obstructing their view, users will naturally move to see the screen clearly. Another potential solution is to direct the user's camera away from their body. Since most users will use their web camera to operate the assistive mouse, this is generally not a practical option. Even so, some users can benefit from using a separate camera.

Overall, deep learning is a better basis for the assistive mouse than image processing. However, neural networks are also more prone to false positives in hand tracking. In other words, they occasionally recognize hands when none are present in the frame. As a result, the cursor will move erratically on the screen. In addition, gesture recognition is performed whenever key points are identified. Due to a false positive hand detection, unexpected mouse actions can be executed. Fortunately, the probability of this occurrence is decreased by the gesture recognition finite state machine.

Chapter 4

Discussion

4.1 Future Work

Haar Cascade Classifiers

Although we discussed the drawbacks of using Haar cascade classifiers for hand detection, we were unable to verify this quantitatively. To do so, a dataset with the supported gestures must first be constructed. Unfortunately, the EgoGesture dataset is insufficient for this purpose. While it can be used to generate key point data, it is less useful for image processing. Since all its images are taken from an egocentric point of view, only the backs of the hands are visible. Haar wavelets detect features such as edges, so a classifier trained on these images will likely not apply to frames depicting the palm and knuckles. Even so, if a suitable dataset existed, it could be possible to perform hand tracking and gesture recognition in a single step. As a result, both the latency and computational resources required to operate the assistive mouse would decrease.

Dynamic Gesture Recognition

Currently, the assistive mouse only supports static gestures. For the majority of gesture recognition models, they are easier to learn. In addition, changes in static gestures are well defined. Despite these advantages, they are not as intuitive as dynamic gestures. For example, it is easier for users to remember swiping up to scroll down than showing three fingers. Therefore, implementing a dynamic gesture recognition model could greatly improve user experience.

However, it also carries additional challenges. More specifically, the system must be able to distinguish dynamic gestures from cursor movement. One potential solution is to make cursor control situational. In other words, it would only be enabled when a specific gesture is performed. While moving the cursor would require additional steps, it would likely be necessary so the hand tracking and gesture recognition modules do not conflict.

Custom Gestures

Another limitation of the assistive mouse is that it only supports a set of predefined gestures. Although they were selected for their simplicity, some users might find them difficult to perform. Similarly, users might prefer alternative methods of human computer interaction. Consequently, the system could be improved by allowing users to define their own gestures. Incorporating them into the existing models could be challenging, but this approach offers better flexibility. Meanwhile, also allowing users to remove unused gestures and decreasing the number of classes could improve the model's accuracy. This would enable users to customize the assistive mouse to fit their needs and abilities.

Speech Recognition

Speech recognition is a relatively unexplored research area for the assistive mouse. While it could be useful for cursor control, it is likely better as an alternative to gesture recognition for mouse actions. For users who lack hand mobility or find many gestures uncomfortable, speech recognition provides another option for them to interact with the computer. Furthermore, it would not suffer from the limitation of moving the palm and cursor to execute an action. Systems such as Apple's Siri and Microsoft's Cortana already enable functionality such as opening applications using voice controls, and speech recognition would extend this further.

Depth Cameras

Depth cameras have become more popular in recent years to provide additional details in problems such as hand tracking. Although some key point detectors estimate this information, depth cameras accurately do so for each pixel in the image. With a greater number of features, more complex models for hand tracking and gesture recognition can be designed. However, these improvements would not be accessible for all users because they require external hardware.

4.2 Conclusion

We present an assistive mouse for human computer interaction built on capturing hand movements. It serves as an alternative system that can help people who experience challenges using a traditional mouse. Through hand tracking and gesture recognition, the same functionality can be realized. Since it only requires a camera, it can easily be used by a wide range of people.

As technology becomes more accessible, it is even more important to develop intuitive systems for human computer interaction. Doing so allows more people to benefit from its capabilities. We believe the assistive mouse represents a step in the right direction towards making computers available for everyone.

Bibliography

[1] John Allen and Richard Xu. "Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces." In: Jan. 2003, pp. 3–7.

[2] Toby Baker, Weihao Dong, Xun Lin, Ayusman Saha, Fangping Shi, and Brian Barsky. *Camera-Based Cursor Control to Increase User Accessibility*. MEng report. EECS Department, University of California, Berkeley, May 2020.

[3] M. Betke, J. Gips, and P. Fleming. "The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 10.1 (2002), pp. 1–10. DOI: 10.1109/TNSRE.2002.1021581.

[4] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[5] J. Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[6] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2019. arXiv: 1812.08008 [cs.CV].

[7] Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang, and Yu-Bo Yuan. "Real-Time Hand Gesture Recognition Using Finger Segmentation". In: *The Scientific World Journal* 2014 (June 2014), p. 267872. ISSN: 2356-6140. DOI: 10.1155/2014/267872. URL: https://doi.org/10.1155/2014/267872.

[8] Amiraj Dhawan and Vipul Honrao. "Implementation of Hand Detection based Techniques for Human Computer Interaction". In: (Dec. 2013). DOI: 10.5120/12632-9151.

[9] International Essential Tremor Foundation. *Facts About Essential Tremor*. 2013. URL: https://www.essentialtremor.org/wp-content/uploads/2013/07/FactSheet012013.pdf.

[10] Frédérick Gianni, C. Collet, and P. Dalle. "Robust Tracking for Processing of Videos of Communication's Gestures". In: *Gesture Workshop*. 2007.

[11] Matilde Gonzalez, Christophe Collet, and Rémi Dubot. "Head Tracking and Hand Segmentation during Hand over Face Occlusion in Sign Language". In: Sept. 2010. DOI: 10.1007/978-3-642-35749-7_18.

[12] R. M. Haralick, S. R. Sternberg, and X. Zhuang. "Image Analysis Using Mathematical Morphology". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.4 (1987), pp. 532–550. DOI: 10.1109/TPAMI.1987.4767941.

[13] *Head Mouse Software for Hands-Free Mouse Control via Web Camera*. URL: https://smylemouse.com/.

[14] 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.

[15] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[16] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552.

[17] Peter Keir, Joel Bach, and David Rempel. "Effects of computer mouse design and task on carpal tunnel pressure". In: *Ergonomics* 42 (Nov. 1999), pp. 1350–60. DOI: 10.1080/001401399184992.

[18] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. *Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks*. 2019. arXiv: 1901.10323 [cs.CV].

[19] Qiong Liu and Guang-zheng Peng. "A robust skin color based face detection algorithm". In: *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*. Vol. 2. 2010, pp. 525–528. DOI: 10.1109/CAR.2010.5456614.

[20] *Move the pointer using head pointer on Mac*. URL: https://support.apple.com/guide/mac-help/move-the-pointer-using-head-pointer-mchlb2d4782b/mac.

[21] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. "GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB". In: *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. June 2018. URL: https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/.

[22] Pradyumna Narayana, Ross Beveridge, and Bruce A. Draper. "Gesture Recognition: Focus on the Hands". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[23] José Manuel Palacios, Carlos Sagüés, Eduardo Montijano, and Sergio Llorente. "Human-computer interaction based on hand gestures using RGB-D sensors". In: *Sensors (Basel, Switzerland)* 13.9 (Sept. 2013), pp. 11842–11860. ISSN: 1424-8220. DOI: 10.3390/s130911842. URL: https://doi.org/10.3390/s130911842.

[24] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. "Realtime and Robust Hand Tracking from Depth". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[25] J. R. Quinlan. "Induction of decision trees". In: *Machine Learning* 1.1 (Mar. 1986), pp. 81–106. ISSN: 1573-0565. DOI: 10.1007/BF00116251. URL: https://doi.org/10.1007/BF00116251.

[26] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[27] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *CoRR* abs/1404.7828 (2014). URL: http://arxiv.org/abs/1404.7828.

[28] Toby Sharp, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, Shahram Izadi, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, and Alon Vinnikov. "Accurate, Robust, and Flexible Real-time Hand Tracking". In: Apr. 2015, pp. 3633–3642. DOI: 10.1145/2702123.2702179.

[29] K Sreedhar. "Enhancement of Images Using Morphological Transformations". In: *International Journal of Computer Science and Information Technology* 4.1 (Feb. 2012), pp. 33–50. ISSN: 0975-4660. DOI: 10.5121/ijcsit.2012.4103. URL: http://dx.doi.org/10.5121/ijcsit.2012.4103.

[30] Ekaterini Stergiopoulou, Kyriakos Sgouropoulos, Nikos Nikolaou, Nikos Papamarkos, and Nikos Mitianoudis. "Real time hand detection in a complex background". In: *Engineering Applications of Artificial Intelligence* 35 (2014), pp. 54–70. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2014.06.006. URL: https://www.sciencedirect.com/science/article/pii/S0952197614001286.

[31] J. Sun, T. Ji, S. Zhang, J. Yang, and G. 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.

[32] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. "A Real-Time Hand Posture Recognition System Using Deep Neural Networks". In: *ACM Trans. Intell. Syst. Technol.* 6.2 (Mar. 2015). ISSN: 2157-6904. DOI: 10.1145/2735952. URL: https://doi.org/10.1145/2735952.

[33] Pretorian Technologies. *Mouse Alternatives for Disabled Users - Assistive Technology*. 2021. URL: https://www.pretorianuk.com/mouse-alternatives.

[34] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[35] Robert Y. Wang and Jovan Popović. "Real-time hand-tracking with a color glove". In: *ACM Transactions on Graphics* 28.3 (2009).

[36] Shiqi Wu, Sihao Chen, Weili Liu, Frank Cai, Yizhou Wang, Xuantong Liu, and Brian Barsky. *Assistive Technology for Navigation, Selection, Pointing, and Clicking in a Mouse-free Environment*. MEng report. EECS Department, University of California, Berkeley, May 2021.

[37] Pei Xu. *A Real-time Hand Gesture Recognition and Human-Computer Interaction System*. 2017. arXiv: 1704.07296 [cs.CV].

[38] Hui-Shyong Yeo, Byung-Gook Lee, and Hyotaek Lim. "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware". In: *Multimedia Tools and Applications* 74.8 (Apr. 2015), pp. 2687–2715. ISSN: 1573-7721. DOI: 10.1007/s11042-013-1501-1. URL: https://doi.org/10.1007/s11042-013-1501-1.

[39] 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. arXiv: 2006.10214 [cs.CV].

[40] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang. "A hand pose tracking benchmark from stereo matching". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 982–986. DOI: 10.1109/ICIP.2017.8296428.

[41] 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.