SimpleGest: Assistive Hand-Gesture Recognition Technology Based on Computer Vision

Fang Hu, Xinying Hu, Yang Huang, Guanghan Pan, and Juntao Peng
The University of California at Berkeley

Cover Summary

The Gesture Mapping / Simplified Gesture Control group consists of five MEng students: Fang Hu, Xinying Hu, Yang Huang, Guanghan Pan, and Juntao Peng[1]. Our audience is the general public, tech generalists, and experts in human-computer interaction, especially those who use computers and with some level of disability and those who are interested in assistive technology. Our goal is to present our motivation, introduce our prototype and its functionality, prove the effectiveness of our system by experiments, and eventually let audiences use our system or know the tech details of it for further research and development purposes.

1 Executive Summary

1.1 Overall Context

People with motor-impairing diseases such as hand dystonia might have difficulty controlling personal computers with a mouse. For them, current control logic as smooth as clicking and dragging may hamper their daily operation of personal computers.

1.2 Team's Accomplishment

We design and implement a system that maps hand gestures to computer input commands with the help of computer-vision. This system potentially frees motor-impaired people from their painful mouse controlling experience.

1.3 Team's Activities

Specifically, our group is divided into three subgroups: simplified cursor control, which focuses on accessibility upgrade on clickable web page elements; cursor-free web browsing, which completely removes mouse and cursor semantics in web browsing; and two-handed gesture control, which controls the computer with combination gesture of two hands.

1.4 Future Work

Our first step is to extend websites that the extension is able to work on. Now our extension works on popular websites such as Google, Baidu, and YouTube, which covers over 90% of search inquiries globally. Although most of the web browsing starts at the search engine page, users' target is other websites. The extension can be improved to handle more websites and make them accessible. Thus, we suggest making the extension compatible with top-hit websites such as Twitter, Facebook, Wikipedia, etc.

Further, expand the browser extension to a desktop application so that simplified cursor control can be used to navigate in all kinds of scenarios when interacting with computers, not limited to web browsing.

To a higher stand, for the greater good of motor-impairment community, we may form an input standard that supports accessibility. Software and websites that comply with the standard will be automatically compatible with our extension

2 Section1

2.1 History of Gesture Command

Probable first idea of gesture computer commands is dated back to the era of first commercialization of computers in 1995. Wexelblat proposed an approach to introduce natural gesture to virtual environments[6]. In this paper, Wexelblat defined what can be deemed as the fundamental pipeline of any gesture controlling system: gesture capturing, semantic extraction, and consumption of such semantic by an appropriate application software.

[1] Permutation is in alphabetical order of surnames.

As time goes by, there were some application and commercialization of gesture command. One of the most successful and welcomed among them is Microsoft Xbox Kinect[7]. The core component, Kinect Sensor, is composed of a regular camera and a set of infrared signal emitter, which is able to sense 3D movement of user. Integrated with Microsoft's family gaming console Xbox, users can send basic control semantics to Xbox console with their gesture with ease.

2.2 Problem definition / Assertion

People with motor-impairment may have difficulty controlling personal computers with a mouse or track pad. Many seemingly simple hand movements, such as clicking and dragging, are almost improbable for them to perform. In order to make computers and other digital devices accessible to them, an alternative set of command input scheme should be designed and implemented. Among many potential amendments, using human gesture as command input is an feasible one, given the advanced computer-vision technologies we possess nowadays.

Graphical User Interface (GUI) is the most intuitive and widely used input method. Cursor makes GUI more efficient and intuitive by providing continuous input in a 2D plane with visual feedback. However, virtual keyboard on screen, either touchable or not, is not as efficient as a physical one. Most mainstream operating systems have built-in onscreen keyboard, such as mouse keys in Windows and accessibility keyboard in macOS, while being rarely used. The primary reason is that physical keyboard enables discrete input with activation threshold (discrete in both time and space) and physical feedback. This feature is especially useful for people with disabilities controlling a cursor.

Inspired by this difference, we focus on defining discrete gesture commands, and stabilize them for error-prone, low-bandwidth input. Cursor Free Web Browsing and Two-handed Gesture Control takes in raw gesture images, extract key information and match with predefined control commands. With gesture translated into computer-understandable control command, Simplified Cursor Control improves web browsing experience by three functions that is adapted to suit slow input with possibly frequent errors.

2.3 Why Is Gesture Command Important?

1. It can help the above group with disabilities achieve human-computer interaction.

2. Offer equal access of digital devices to those with limited body control, and thus promotes diversity, equity, inclusion, and belonging of the society.

2.4 Existing Solution and their shortage: (State of the arts)

Betke et al. presents the "Camera Mouse" system for people with disabilities that can track people's movement via a video camera and convert it into mouse movement to control the computer [1]. The paper demonstrated success on people with great disability. People with disability are able to spell characters and move mouse cursor with the help of their system. While their approach target primarily on people with severe disability, our project focus on people with motor-impaired disease. The goal of our project is not only help those people to spell characters and move cursor, but also simplify more complex tasks such as video control and web browsing, and eventually use the computer like ordinary people.

Bubble cursor is an adaptive cursor that expands selection area, different from a usual cursor with only one pixel selection area [3]. It dynamically changes activation area and acquires target by a ranking based on proximity of surrounding elements, more specifically, by locating nearest element. Empirical studies show that bubble cursor significantly outperforms ordinary point cursor by and can be accurately modeled and predicted using Fitts' law. However, formulas derived in the paper are based on oversimplification of actual scenario, that is, only round elements are taken into consideration. In real-world scenarios, selectable elements come with all kinds of shapes, mostly rectangle. The effect of bubble cursor remains unverified for other shapes of elements as it alters the formula of selection time and error rate. Besides, experiments were carried out on a simulated environment, which cannot fully prove its usability and efficiency in widely-used operating system by general computer users. Lastly, the original paper only serves as a proof-of-concept and does not include any usable software systems. No existing software is built well enough to demonstrate the power of bubble cursor.

Mankoff et al. lists several standards for low bandwidth in web accessibility in their paper [5]. They demonstrated two tools to comply with the standards: a modified web browser and a proxy that modifies HTML. However, their approach is limited to the web-browsing scenario, and mainly targeted on low bandwidth users who can only produce one or two signals when communicating with a computer. In addition, since the paper was written in 2002, their proposed solution only works with static websites, while a large portion of current web pages are dynamic. In our project, we extend the idea introduced in the paper from web browsing to all human-computer interaction scenarios, and also provide solution for people with disability to browse through dynamic web pages.

Grif et al. presents a mouse cursor control interface

to the end of human-computer interaction [2]. The implementation is based on low-cost webcams, which have become widely available with adequate image quality in recent years. The system captures users' hand with fingertips marked with color strips on a hand pad as background, and detects finger strips color by fluctuation of illumination technique to enhance recognition accuracy. The system can convert hand gestures to three mouse events, left click, right click, and double click. Results show good performance of the system in poorly-lit environment. The use of color strips on fingertips and a hand pad significantly restricts its usage scenarios. Users cannot rotate hand on the plane of the hand pad. Mouse events the system is able to recognize are also limited. No cursor movement, page scrolling, or other complex tasks can be performed using the system. Besides, experiments are done in a dedicated gesture recognition software with no actual computer control scenarios.

2.5 Our solution

Luckily, we may take advantage of computer vision technology to help people with difficulties using the mouse as a computer input.

3 Section 2

3.1 Simplified Cursor Control

To make cursor control suitable for low-bandwidth, inaccurate input, we present Simplified Cursor Control. It simplifies controlling through reduction of cursor choices and cursor movement. By reducing choices, we limit the possibility of errors by inaccurate input because there are fewer elements or mouse events a gesture can perform. We also increase activation threshold (space - insensitive to travel distance). We also achieve the same level of hyperlink selection flexibility by letting the cursor auto focus on nearest clickable element. This significantly reduces the distance cursor needs to travel and effectively increases the input bandwidth.

Our system is intuitive, error-tolerant, and efficient by increasing the bandwidth. A drawback is it still requires some level of cursor control. Also, nearest element clicker is unable to perform text selection.

For deliverables, we have both Chrome extension and Firefox add-on. The former is available by loading packed/unpacked extension source code, and the latter can be downloaded on Firefox Add-ons store for free.

Hyperlink enlarger Hyperlink enlarger highlights and enlarges the clickable area of a search result block when hovering.

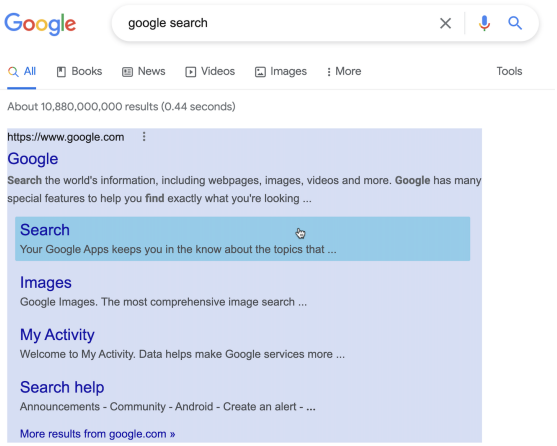

Figure 1: Clickable area of searched results are enlarged to the highlighted area with hyperlink enlarger

It works on a results page after a search operation carried out in a search engine as shown in figure 1. By changing the attribute of the box wrapping a search item in HTML, we makes the box clickable with redirection to target page, instead of just the hyperlink itself. This results in a four times larger . According to Fitt's law, this will increase the ease of cursor control by a log of 4 times.

Currently, it works on Google search results page. Permission to read the page is needed.

Button Enlarger The button enlarger feature makes a button element larger when hovering. The feature works on any page with dense button layout, which makes it hard to locate the mouse cursor on the desired button, especially When using low bandwidth method to control cursor. By enlarging the hovered button and the clickable area, low level errors are captured, hence reducing the possibility of inaccurate misclick.

Nearest Element Clicker Nearest element clicker highlights the nearest clickable hyperlink and makes it clickable without hovering. Nearest element is defined as the element which has the smallest center-to-cursor distance. For example, the center of a rectangle is its geometrical center, i.e., the midpoint of the diagonal end points of rectangle.

It keeps scanning HTML file in the page source looking for the nearest elements with hyperlinks.

Current distance definition could be, however, less intuitive for clickable elements separated by line ends, resulting in longer time for locating correct elements. Another drawback is CPU load of continuous scan after mouse move event.

3.2 Cursor-Free Web Browsing

The key point in cursor-free web browsing is to solve the cursor-instability problem caused by the unconscious muscle twitches. These vibrations caused by the incessant minor changes of hand gesture, would lead to a unstable cursor and even trigger mis-classifications when we try to extract and analyze gesture from video streams. Therefore, we aim to replace an unstable cursor with a more robust gesture semantics analysis system.

Server-Client Communication Unlike other gesture control methods that bind hand movement to a moving cursor, our approach decided to completely discard cursor, but design a new gesture operation system. This implies that we need to explore a method to transmit semantic messages to and fro. Due to the fact that the control of a computer and the capture of a human gesture is in real-time, a stable communication channel should be established between MediaPipe and our browser extension.

WebSocket is a suitable application layer communication protocol that supports two-way full-duplex exchange of information. Its low overhead and persistence makes it the ideal communicator of our Chrome extension. We use a python package called Flask that realizes its own version of WebSocket.

In order to successfully send message between extension and MediaPipe. We divide our implementation into three parts (processes): python MediaPipe client, python Flask WebSocket server, and Chrome extension. The MediaPipe client captures gestures and then extracts control points from the video stream. The Flask server receives control point messages from MediaPipe client in the form of JSON, and then forwards these messages to the Chrome extension. The Chrome extension receives these messages and maps them to corresponding JavaScript code segments that performs actual browser actions.

TODO: ADD A FIGURE OF THREE PARTS

Detection of Clickable Elements One of the major obstacle of removing cursor is the fact that almost all of the current websites are designed in a way that requires extensive click events. However, with the cursor-free requirement, how can we perform click on HTML elements? The answer is fairly straightforward: use JavaScript to call elements' click function to trigger the click events.

Now, the problem is converted to find out every clickable HTML element in a web page. As a matter of fact, there are three categories of clickable elements in a HTML web page that resides in a modern JavaScript browser engine: default clickable HTML elements (such as hyperlink and button), HTML tags that are declared with a onClick function, and JavaScript objects with onClick member function. The capturing of first two categories is fairly simple, we can just iterate through the HTML elements of a web page and test whether it has a onClick function. The third category, on the other hand, is quite tricky to detect as their onClick member functions are dynamically set during runtime.

We achieve this by a hack that adds a stub in the onClick setter of HTML elements' base class. The stub we add will automatically append current element's ID after a global array so that when the third category elements inherited from this base class at runtime, we are able to keep track of who is setting its onClick member function.

After capturing all clickable elements, the rest of our work is quite simple. By calling each element's onClick function, we simulate a click event in the browser engine. Now, we successfully trigger a click command without a single presence of mouse cursor!

TODO: Add a figure of clickable selection.

Control Point-Based Hand Gesture Command Set TODO: Introduce our gesture command set and how we detect them.

3.3 Two-Handed Gesture Control

Two handed- gesture control allows users to give computers real-time commands such as cursor control, clicking, volume control etc. only by showing predefined hand gestures in front of the web camera. The system, shown at Fig x, consists of two main parts: real-time key-point detection, gesture recognition and gesture-command mapping, which will be detailedly demonstrated below.

Keypoint Detection In order to track the user's hand movement accurately and in real-time, we used Mediapipe solution to capture user's hand keypoints for the later gesture recognition algorithm. Mediapipe, which is based on machine learning models and opencv, has the ability of two-hand detection from a single frame, predicting 21 3D keypoints for each hand. An example of keypoint extraction is shown in Fig x. The relative position of each keypoint will be kept for gesture recognition computation. For now, only 2D keypoint information is utilized.

Gesture Recognition In this project, two non-ML based gesture recognition algorithms are implemented: relative keypoint position and template matching. Based on two mechanisms, seven hand gestures have been defined for the system, shown in Fig x.

Relative Keypoint position TODO

Template Matching method Template Matching method compares the keypoints detected from the streaming frame and the pre-extracted keypoint positions from loaded hand gesture templates, and categorizes the user's current position to the predefined gesture with the highest similarity. The following steps describe the process to achieve the result, also shown in Fig x.

First, we keep several images of different gestures as the template, or "ground truth" of the system. The extracted keypoints are save as CSV files for later use.

When the program rans, each frame is used to extracted keypoints (30 FPS during our experiment). In order to remove the effect of scale, shiftness, and rotation, the extracted keypoints from the current frame are applied to the affine transformation between each of the template gesture keypoints.

After the affine transformation, error computed between the transformed current frame keypoints and template gesture keypoints shows their similarity, or the probability of the current frame being categorized as the corresponding template. The less the error shows, the higher the score the template gesture receives. By comparing all of the scores, the gesture with highest score above the threshold will be selected and recognized.

Two-handed Control Flow The gesture recognition function is then mapped to various computer commands. To make the most use of the variety of two-handed gestures, we developed a two-handed gesture control scheme to for different set of commands and operation, shown in Fig x.

References

[1] BETKE, M., GIPS, J., AND FLEMING, P. The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on neural systems and Rehabilitation Engineering 10*, 1 (2002), 1–10.

[2] GRIF, H.-S., AND FARCAS, C. C. Mouse cursor control system based on hand gesture. *Procedia Technology 22* (2016), 657–661.

[3] GROSSMAN, T., AND BALAKRISHNAN, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), pp. 281–290.

[4] KOETSIER, J. Intel scientist who built stephen hawking's communicator is updating it with ai and gpt-2, open sourcing it, Aug 2020.

[5] MANKOFF, J., DEY, A., BATRA, U., AND MOORE, M. Web accessibility for low bandwidth input. In *Proceedings of the fifth international ACM conference on Assistive technologies* (2002), pp. 17–24.

[6] WEXELBLAT, A. An approach to natural gesture in virtual environments. *ACM Trans. Comput.-Hum. Interact. 2*, 3 (sep 1995), 179–200.

[7] ZHANG, Z. Microsoft kinect sensor and its effect. *IEEE MultiMedia 19*, 2 (2012), 4–10.