Night Fox: A Communication Assistant Application for Paralyzed People based on Eye tracking and Sketch Recognition

Qiancheng Liu, Aniket Bonde, Bing Jiang, Chao Gu

Abstract

A communication assistant application for paralyzed people is studied and evaluated. The application is based on normal webcam and totally written in JavaScript, which make this application available for every web user. The WebGazer.js API is applied for data collection. Collected eye-gazing data is processed and recognized to locate the object user want to choose, while using Rubine features and template matching algorithm. This application could help user to achieve certain purpose and the recognition accuracy for the application is about 60% during the evaluation.

Keywords Sketch Recognition, Eye tracking

1 Introduction

1.1 Motivation

Paralyzed patients are unable to communicate with other people easily. Patients paralyzed below the neck could caused by trauma reason. They could talk but not able to use their hands. For ALS patients, the only part of their body they could control are their eyes. Different modern technologies are developed to help paralyzed people to have a better life experience, such as Neural Implant, EEG based BCI, and eye tracking. Neural Implant has surgical risk and may be not for every patient. EEG based BCI with huge training could help people to control a mechanical arm. But it could not help them to communicate with people. Eye tracking provides one great opportunity to help them.

1.2 Domain related

Our project is related with eye tracking and sketch recognition. The system is to determine the object that user is tracking with eyes. So we need to eye tracking system to record users' eye movements and to recognize and predict users' eye movement path. Based on the eye movement path and the real object moving paths models, we need a sketch recognition system to match eye movement path with object moving path model. So our project is the combination of eye tracking and sketch recognition.

1.3 Problem we are solving

We are trying to help paralyzed people to have a better life experience. We want to let people select moving object just using eyes. We want to build an application for the paralyzed people or patients who can not talk or move their hand easily to communicate with others. The application has an interface that has items moving around following different paths, and users can select an item by following the item on the screen. We get the track of their eye movement, and using some sketch recognition techniques, we can recognize which item he is following. Being followed means being selected. After detecting the selected item, the application will make corresponding response according to user's need.

1.4 Solution

We want to build an application to solve our problem. This application can tracking user's eye movement path when user is staring at a specific moving object. After preprocessing the eye movement path data, this application will try to match the eye movement path with object movement path, and to determine which object the user is staring at. Then the screen will show the object user is tracking. And the system will let user to confirm whether it is the object user is tracking by offering two buttons.

2 Related works

2.1 Technical References

Our project uses the library WebGazer that was presented by Alexandra Papoutsaki, 2016. WebGazer is a new approach for scalable and self-calibrated eye tracking on the browser using webcams. WebGazer can be added on any webpage and aims to democratize eye tracking by making it accessible to the masses. It showed that incorporating how gaze and cursor relate can inform a more sophisticated eye tracking model. Using the best of the 3 tracking libraries and with the best regression model, the mean error is 104 pixels in a remote online study.

In its current state, WebGazer is useful for applications where the approximate location of the gaze is sufficient. The best arrangement for WebGazer mimics the abilities of a consumer-grade eye tracker, but can be deployed on any website. Its utility will improve as devices gain more powerful processing capabilities and webcams capture better images in poor lighting. It is a big step towards ubiquitous online eye tracking, where scaling to millions of people in a privacy-preserving manner could lead to innovations in web applications and understanding web visitors.

2.2 Peripheral References

The study of eye movements pre-dates the widespread use of computers by almost 100 years (for example, Javal, 1878/1879). Beyond mere visual observation, initial methods for tracking the location of eye fixations were quite invasive — involving direct mechanical contact with the cornea. Dodge and Cline (1901) developed the first precise, non-invasive eye tracking technique, using light reflected from the cornea. Their system recorded only horizontal eye position onto a falling photographic plate and required the participant's head to be motionless. Shortly after this, Judd, McAllister and Steel (1905) applied motion picture photography in order to record the temporal aspects of eye movements in two dimensions. They recorded the movement of a small white speck of material inserted into the participants' eyes rather than light reflected directly from the cornea. These and other researchers interested in eye movements made additional advances in eye tracking systems during the first half of the twentieth century by combining the corneal reflection and motion picture techniques in various ways (see Mackworth & Mackworth, 1958 for a review).

In the 1930s, Miles Tinker and his colleagues began to apply photographic techniques to study eye movements in reading (see Tinker, 1963 for a thorough review of this work). They varied typeface, print size, page layout, etc. and studied the resulting effects on reading speed and patterns of eye movements. In 1947

Paul Fitts and his colleagues (Fitts, Jones & Milton, 1950) began using motion picture cameras to study the movements of pilots' eyes as they used cockpit controls and instruments to land an airplane. The Fitts et al. study represents the earliest application of eye tracking to what is now known as usability engineering — the systematic study of users interacting with products to improve product design.

Around that time Hartridge and Thompson (1948) invented the first head-mounted eye tracker. Crude by current standards, this innovation served as a start to freeing eye tracking study participants from tight constraints on head movement. In the 1960s, Shackel (1960) and Mackworth and Thomas (1962) advanced the concept of headmounted eye tracking systems making them somewhat less obtrusive and further reducing restrictions on participant head movement. In another significant advance relevant to the application of eye tracking to Human–computer interaction, Mackworth and Mackworth (1958) devised a system to record eye movements superimposed on the changing visual scene viewed by the participant.

Eye movement research and eye tracking flourished in the 1970s, with great advances in both eye tracking technology and psychological theory to link eye tracking data to cognitive processes. See for example books resulting from eye movement conferences during this period (i.e., Monty & Senders, 1976; Senders, Fisher & Monty, 1978; Fisher, Monty & Senders, 1981). Much of the work focused on research in psychology and physiology and explored how the human eye operates and what it can reveal about perceptual and cognitive processes. But publication records from the 1970s indicate a lull in activity relating eye tracking to usability engineering. We presume this occurred largely due to the effort involved not only with data collection, but even more so with data analysis. As Monty (1975) puts it: "It is not uncommon to spend days processing data that took only minutes to collect" (pp. 331–332). Work in several human factors/usability laboratories (particularly those linked to military aviation) focused on solving the shortfalls with eye tracking technology and data analysis during this timeframe. Researchers in these laboratories recorded much of their work in US military technical reports (see Simmons, 1979 for a review).

Much of the relevant work in the 1970s focused on technical improvements to increase accuracy and precision and reduce the impact of the trackers on those whose eyes were tracked. The discovery that multiple reflections from the eye could be used to dissociate eye rotations from head movement (Cornsweet & Crane, 1973) increased tracking precision and also prepared the ground for developments resulting in greater freedom of participant movement. Using this discovery, two joint military/industry teams (US Airforce/Honeywell Corporation and US Army/EG&G Corporation) each developed a remote eye tracking system that dramatically reduced tracker obtrusiveness and its constraints on the participant (see Lambert, Monty & Hall, 1974; Monty, 1975; Merchant et al., 1974 for descriptions). These joint military/industry development teams and others made even more important contributions with the automation of eye tracking data analysis. The advent of the minicomputer in that general timeframe provided the necessary resources for high-speed data processing. This innovation was an essential precursor to the use of eye tracking data in real-time as a means of Human–computer interaction (Anliker, 1976). Nearly all eye tracking work prior to this used the data only retrospectively, rather than in real time (in early work, analysis could only proceed after film was developed). The technological advances in eye tracking during the 1960s and 1970s are still seen reflected in most commercially available eye tracking systems today (see Collewijn, 1999 for a recent review).

Psychologists who studied eye movements and fixations prior to the 1970s generally attempted to avoid cognitive factors such as learning, memory, workload, and deployment of attention. Instead their focus was on relationships between eye movements and simple visual stimulus properties such as target movement, contrast, and location. Their solution to the problem of higher-level cognitive factors had been "to ignore, minimize or postpone their consideration in an attempt to develop models of the supposedly simpler lower-level processes, namely, sensorimotor relationships and their underlying physiology" (Kowler, 1990, p. 1). But this attitude began to change gradually in the 1970s. While engineers improved eye tracking technology, psychologists began to study the relationships between fixations and cognitive activity. This work resulted in some rudimentary, theoretical models for relating fixations to specific cognitive processes (see for example work by Just & Carpenter, 1976a, 1976b). Of course scientific, educational, and engineering laboratories provided the only home for computers during most of this period. So eye tracking was not yet applied to the study of human–computer interaction at this point. Teletypes for command line entry, punched paper cards and tapes, and printed lines of alphanumeric output served as the primary form of Human–computer interaction

As Senders (2000) pointed out, the use of eye tracking has persistently come back to solve new problems in each decade since the 1950s. Senders likens eye tracking to a Phoenix raising from the ashes again and again with each new generation of engineers designing new eye tracking systems and each new generation of cognitive psychologists tackling new problems. The 1980s were no exception. As personal computers proliferated, researchers began to investigate how the field of eye tracking could be applied to issues of human–computer interaction. The technology seemed particularly handy for answering questions about how users search for commands in computer menus (see, for example, Card, 1984; Hendrickson, 1989; Altonen, Hyrskykari & Räihä, 1998; Byrne et al., 1999). The 1980s also ushered in the start of eye tracking in real time as a means of human–computer interaction. Early work in this area initially focused primarily on disabled users (e.g., Hutchinson et al., 1989; Levine, 1981, Levine, 1984). In addition, work in flight simulators attempted to simulate a large, ultra-high resolution display by providing high resolution wherever the observer was fixating and lower resolution in the periphery (Tong & Fisher, 1984). The combination of real-time eye movement data with other, more conventional modes of user–computer communication was also pioneered during the 1980s (Bolt, 1981, 1982; Levine, 1984; Glenn et al., 1986; Ware & Mikaelian, 1987).

In more recent times, eye tracking in human–computer interaction has shown modest growth both as a means of studying the usability of computer interfaces and as a means of interacting with the computer. As technological advances such as the Internet, e-mail, and videoconferencing evolved into viable means of information sharing during the 1990s and beyond, researchers again turned to eye tracking to answer questions about usability (e.g., Benel, Ottens & Horst, 1991; Ellis et al., 1998; Cowen, 2001) and to serve as a computer input device (e.g., Starker & Bolt, 1990; Vertegaal, 1999; Jacob, 1991; Zhai, Morimoto & Ihde, 1999). We will address these two topics and cover their recent advances in more detail with the separate sections that follow.

3 Approach

3.1 User Interface

The system has a hierarchy structure that allows users to make their selection step by step. And to make more room for the moving objects, the interface is simplifies to the minimal number of elements that is necessary for the selection. Story board of the user interface is shown below.

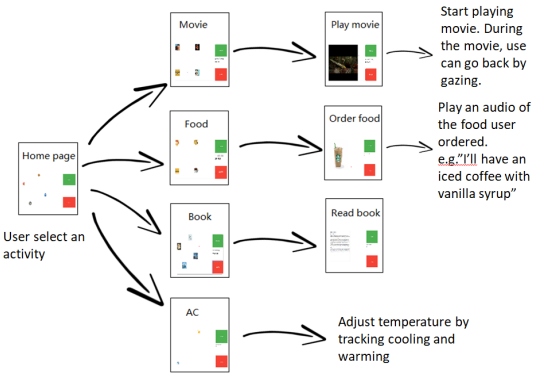

Figure 1. Storyboard of the system

We have two kinds of UIs: directory UI and final UI as they are the final layer of the system. Below is the wireframe of these two UIs.

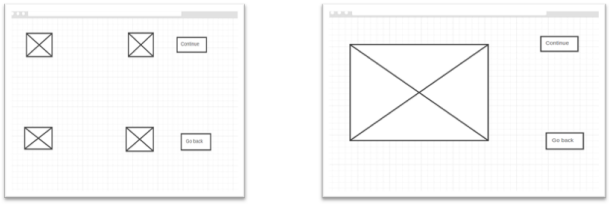

Figure 2. wireframe of directory UI and final UI

3.1.1 UI Directory

Directory UI lists all of the paths that help user to find the item user want. In directory UI, the left screen displays four moving object, and the right screen displays two buttons. Below is a wireframe of directory UI.

Figure 3. Screenshot of home page

In the home directory page, it shows four moving object representing four categories: Food, Movie, Book and Air Condition.

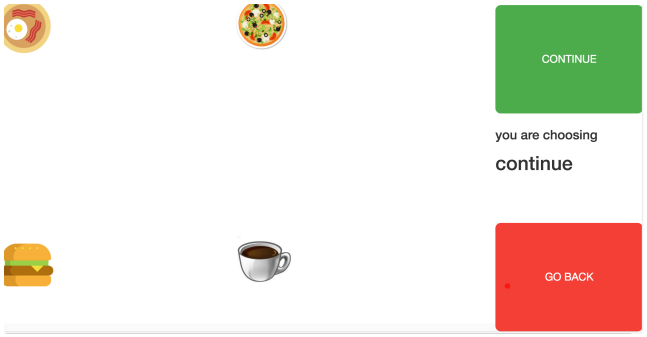

Figure 4. Screenshot of food selection

In the food directory page, it shows four moving object representing four kinds of food: Pizza, Coffee, Hamburger and Breakfast.

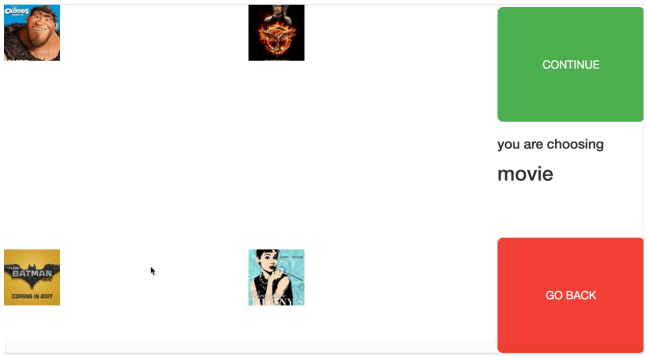

Figure 5. Screenshot of movie selection

In the movie directory page, it shows four moving object representing four movies.

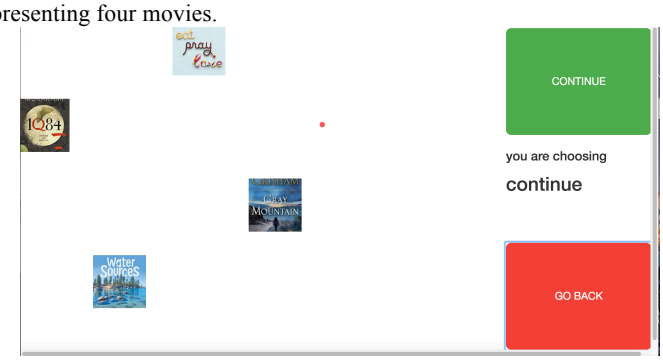

Figure 6. Screenshot of book selection

From the book directory page, it shows four moving object representing four books.

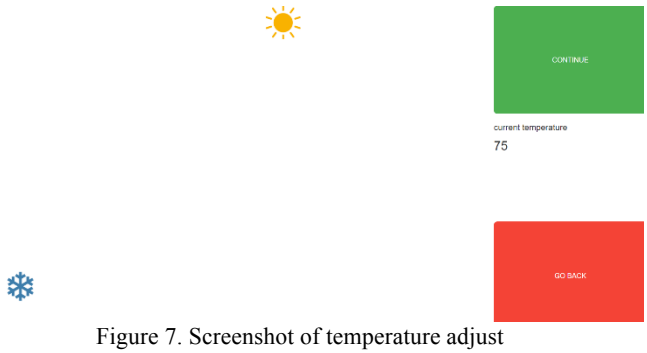

Figure 7. Screenshot of temperature adjust

From the air conditioning directory page, it shows two moving object representing increasing temperature and decreasing temperature.

3.1.2 Final UI

When the user finally find the item form directory UI, it goes to the final UI. Suppose the user want Mcdonald's food from directory UI, the system will render an image of the food user ordered, alone with the audio playing the name of the food.

Figure 8. Screenshot of final UI for food

3.2 Interaction

In the directory UI page, the left screen displays four moving object and user just stare at one moving object. The system will recognize the object that user is tracking.

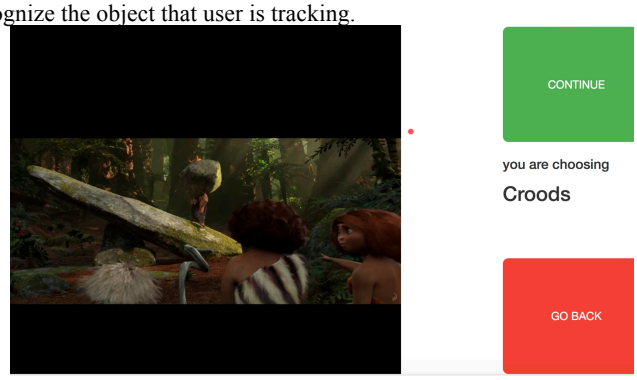

Figure 9.Screen shot when the system is playing movie

And then it goes to the object that system predicted and stay on that page to let user confirm is this the right page. User can stare at one of the two buttons on the right screen: Confirm and Go Back. If user confirm that this is the right item user want, e.g. Batman movie, then UI will play Batman movie; if user choose go back option, then the system will go back to the previous UI.

3.3 Justify User Interface and Interaction Design

Our web based application let user to choose the object by tracking moving object in the left screen and system will predict the object user want and the system goes to the next level directory UI or final UI. And then user will stare one of the two buttons to confirm the correct prediction or to choose go back option.

3.4 Entire Interaction Session

First of all, user needs to train the system by staring at the point user clicking using mouse. It may take 20 to 40 seconds. Then the system shows the home directory page. In the home directory page, it shows four moving object representing four categories: Food, Movie, Book and Air Condition. If user choose Food object, then the system will go to the Foot directory UI. User may confirm this is the correct one. And now this page shows four moving object representing four kinds of food: Pizza, Coffee, Hamburger and Breakfast. User may stare at the moving coffee object, and then system may mismatch user's eye movement, and it go to the Pizza page. And user stare at the go back button, it goes back to the food directory UI. This time the system correctly predict the coffee object, and then it goes to the coffee UI page.

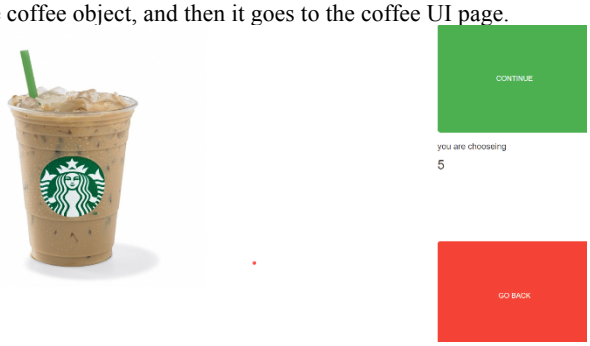

Figure 10.Screen shot when the system is playing audio for coffee

User confirm this is the right one he/she wants, then this information will transferred to the appropriate machine that will response user's choice. In our app, we can only response to limited choices, like a book, UI will display the content of that book. If it is a movie, UI will play that movie.

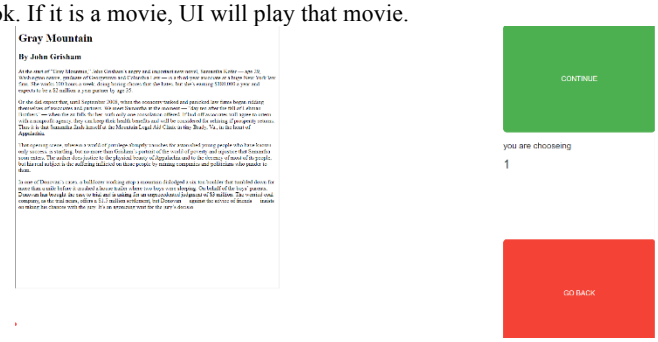

Figure 11. Screenshot when reading books

For the air conditioner part, it doesn't have a final UI, instead, it has a response on the right side between two confirm buttons showing current temperature.

4. Recognition Systems

4.1 Library Introduction

WebGazer.js is an eye tracking library that uses common webcams to infer the eye-gaze locations of web visitors on a page in real time. The eye tracking model it contains self-calibrates by watching web visitors interact with the web page and trains a mapping between the features of the eye and positions on the screen. WebGazer.js is written entirely in JavaScript and with only a few lines of code can be integrated in any website that wishes to better understand their visitors and transform their user experience. WebGazer.js runs entirely in the client browser, so no video data needs to be sent to a server. WebGazer.js runs only if the user consents in giving access to their webcam.

4.2 Training and Data Collection

4.2.1 Training

Before the recognition starts, a training procedure is applied for every individual. The training page is shown below.

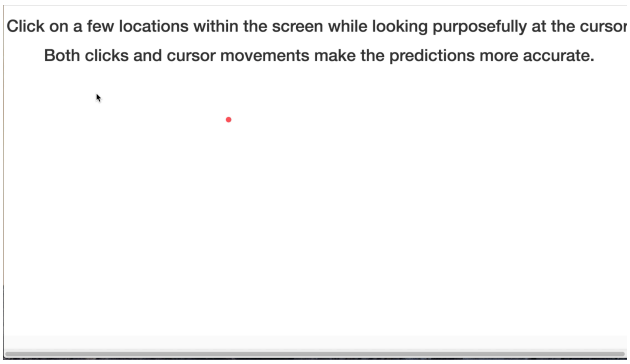

Figure 12. Training the eye tracker

During the training, users should sit in a proper distance from the webcam and keep their faces still.

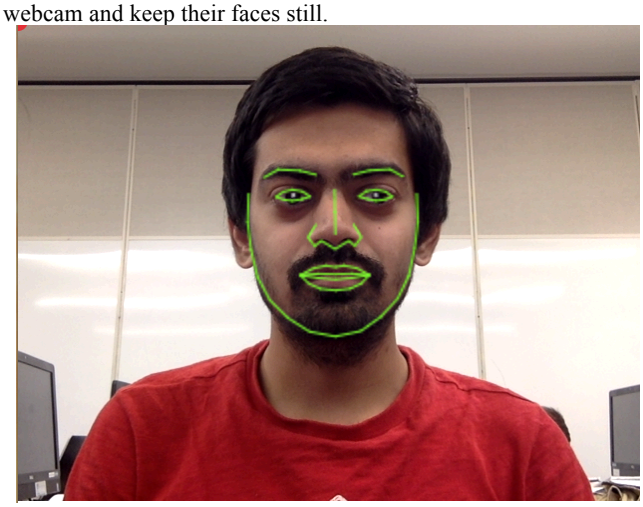

Figure 13. Eye tracking is locating user's eyes

Users should move and click mouse when they are staring at cursor. The whole training section would last at least 40 seconds. The red dot on the screen represents the predicted eye-gaze location by WebGazer.js. After the training section, the data collection starts for the eye-sketched recognition.

4.2.2 Data Collection

Data is continually collected by WebGazer.js every 10 milliseconds. Collected data were considered as stroke points about every 1.5 seconds.

Each eye-gazed stroke would be processed and recognized independently. Every 5 recognized strokes would form a sketch for recognition later.

4.3 Recognition Algorithms and Features

4.3.1 Preprocess

When the recognition system gets the eye movement path data from WebGazer.js, it preprocesses the data. It calculates the mean values of the x and y coordinates of the start and end 15 points of the stroke.

4.3.2 Features

We select some of the Rubine features for the sketch recognition. We use the cosine between the start and end points, sine between the start and end points to detect eye movement including, right, left, up, down. Counter for hitting button areas "Yes&No" is another feature we utilized for detect eye-gazing button.

4.3.3 Algorithms

Our system recognizes five strokes from sketch. And template-matching algorithm is applied for sketch recognition on this eye-traced sketch to determine which item user was following or which button user was gazing at.

We use the Nomenclature as follows:

R - Moving right
L - Moving left
U - Moving Up
D - Moving Down
N - Hitting "No" more than "Yes"
Y - Hitting "Yes" more than "No"

Hamming distance between sketches and templates were calculated. Template is setted as follows:

['D','R','U','L','D'] -> Item 1
['L','D','R','U','L'] -> Item 2
['R','U','L','D','R'] -> Item 3
['U','L','D','R','U'] -> Item 4
['Y','Y','Y','Y','Y'] -> Correct
['N','N','N','N','N'] -> Incorrect

For instance, the hamming distance for sketch ['L','L','U','L','D'] are listed as following:

Item 1 -> [0,0,1,1,1] -> Score = 3 (winner)
Item 2 -> [1,0,0,0,0] -> Score = 1
Item 3 -> [0,0,0,0,0] -> Score = 0
Item 4 -> [0,1,0,0,0] -> Score = 1
Correct -> [0,0,0,0,0] -> Score = 0
Incorrect -> [0,0,0,0,0] -> Score = 0

4.4 Recognition System Hierarchy

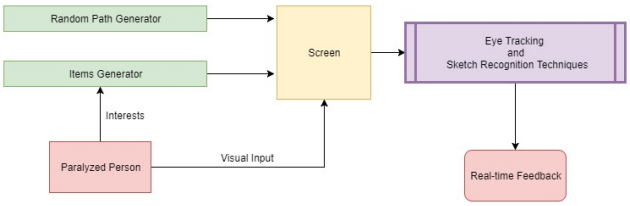

Figure 14. Workflow of our proposed project

We have two separate eye tracking system: Dynamic Object Tracking System and Static Object Tracking System. The architecture of the Dynamic Object Tracking System uses WebGazer.js library to track user's eyes when objects flying in the screen. At the same time, backend processes the eye movement data, and match eye tracking with object moving path and thus recognize the object that user's eyes are tracking. The architecture of the Static Object Tracking System is similar to the Dynamic Object Tracking System, the different is that Dynamic Object Tracking System is to recognize which object user is tracking, and Static Object Tracking System is to confirm whether the recognition is correct or not.

System working procedure:

Step 1: Multiple objects moving on the screen with designed paths and user's eyes are tracking only one of these objects.

Step 2: Webcam is catching user' eye movement and transfer movement data to WebGazer.js.

Step 3: WebGazer.js parses eye movement data and predicts tracking points, and then transfer predicted data to Sketch Recognition System

Step 4: Sketch Recognition System analyzes predicted points data and recognizes tracking path and matches with one of the designed paths. Then it transfers the designed path id to the client side webpage.

Step 5: Client browser shows user the predicted object to user, also it offers options to user to confirm whether the prediction is correct or not with two static object: Correct or Incorrect. User can stare at one of the project, and webcam with catch user eye movement.

Step 6: Similar to Step 1 to Step 4, but just two static object on the screen, it will recognize user' confirm: Correct or Incorrect.

Step 7: If the recognition is incorrect, then go to Step 1; if correct, then it works.

4.5 Data and Metadata for Recognition Purpose

The data our application used is in-person user training data. During the training process, the webcam would capture data via log files from the Tobii EyeX eye tracker and server logs for WebGazer.js. Both datasets were parsed down to a time series of predictions and grouped into 100 millisecond bins. The error was computed as the average distance between each regression and Tobii EyeX for the equivalent bin. Tracking.js data were excluded due to performance issues running both eye trackers simultaneously. The collected data are considered as stroke point for about 1.5 seconds.

5. Evaluation

5.1 User Study Design

The whole design was tested by all group members and randomly selected 10 participants. Sketch recognition part is tested independently on local server. During the experiment, no pre-trained data was used. The user's task includes first train eye tracker and then get familiar with the system.

We inserted the "alert information" after each phase so we can take notes of the result, then the system continues. But users are not allowed to use mouse or keyboard.

Our evaluation can be divided into two parts: sketch recognition and system performance.

5.2 Evaluation 1: Sketch Recognition

This part mainly focuses on the accuracy of sketch recognition. There are two different kinds of sketches: eye movement and eye gazing.

1. Recognition accuracy for eye movement

The experiment is done under ideal environment, user is following an item, and the system will return an array of directions with alert information. Then we will take notes of the array returned and the actual movement. Then we calculate the accuracy for each different direction. We collected 4 different directions for 50 experiments, which is 200 sketches in total, and the accuracy is shown in table 1.

Direction	Accuracy
Left	76%
Right	72%
Up	60%
Down	70%

Table 1. eye movement accuracy

2. Recognition accuracy for Gaze

The experiment is done under ideal environment, User stare at a button, and the system will return an array of 'Y' or 'N'. Then we take notes and manually count the the accuracy.

We had 50 experiments for 'Y' and 'N' seperatly, accuracy is shown in table 2

Gaze	Accuracy
Continue	82%
Go back	64%

Table 2. Gazing accuracy

5.3 Evaluation 2 : System Performance

1. Movement speed:

Items movement speed can greatly affect the accuracy. To find the suitable movement speed, we carried out an experiment. We randomly selected a user to test the system when items are running at different speed. During the experiment, the user task is to select the required items with their eyes. If the user can successfully select the required item without going back, then we count it as accurate. Accuracy is shown in Figure 15.

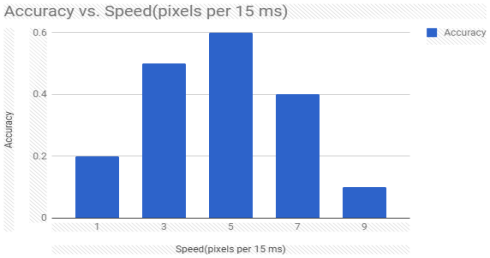

Figure 15. Total accuracy at different speed

2. Number of strokes

Number of strokes also influence the accuracy. Similarly, we randomly select a user to test the system when of items are running at different number of strokes. Figure 16 is the accuracy at different number of strokes.

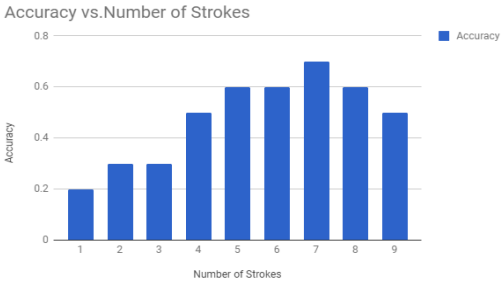

Figure 16. Total accuracy at different number of strokes

3. Environment

Since the eye tracker is based on web camera, so without doubt, the environment will have great influence on the accuracy of the recognition. Yet, we still want to know exactly how big the difference will be, so we carried out an experiment. We randomly select 10 users to use our application. They are asked to perform two experiments. Accuracy is counted by the number of times success without going back.

- First experiment is under noisy environment (Annex library, not bright light, wear glasses, using laptop)

- Second experiment is in ideal environment, background. (Good lighting,no glasses, a better web camera, good distance in front of camera)

Users task is to select the required items with their eyes. Accuracy is shown in table 3.

Experiment	Accuracy
Noisy	50%
Ideal	70%

Table 3. Accuracy under different environment

6. Discussion

6.1 Discuss Raw Result

Training time is an important factor that would affect the accuracy. During the training process, the WebGazer has user's basic model and train the user's eye movement and camera catching data. It helps to decrease the noises and remove some other disturbances. If the training time decreases, WebGazer doesn't have enough data to make the user model better and there would be larger error of the eye movement tracking path. That would affect the accuracy of object matching process. But if it takes much time training, the user may be impatient. After the trade off, we determine to have a training time of 40 seconds.

The position of the button on the screen great influence on accuracy, if the button on the right side of the screen and the distance between the buttons is shorter, then the accuracy would decrease.

The speed and strokes of the moving objects has great influence on accuracy. If the moving speed is to a little slow, the accuracy can be ensured. If the speed is a little bit fast, the accuracy decreases significantly. That's because there is a delay for the eye-tracking process. There are too many eye movement points need to be preprocessed. If too small number of points are selected, the time is reduced by the accuracy also decreases. It also takes some time to process the eye movement path points and sketch recognition system for template matching. Thus the object moving speed cannot be fast.

Keep your eyes large open can help improvement the accuracy. If a user has a large size eyes, it helps to increases the accuracy. That's because when the system is tracking user' eyes, it has to distinguish user' eyes and face. If user has a smaller eyes or user doesn't keep his/her eyes large open, the system cannot clearly detect the eye movement and thus it is difficult to tracking user' eye movement path and thus the object matching accuracy will decrease.

Similarly, the distance between the user and camera has huge effect on the accuracy. Long distance will increase the difficulty to catch eye movements and increase noises. Also longer distance means smaller eye movement angles. In order to ensure the accuracy of eye tracking, it's better to keep the eyes within 40 cm.

Also, when user using our application, it's better to let user keep user's head or face static. If user keep shaking his/her head or to make so many emotions, it would decrease the accuracy. Because WebGazer use user's face as the coordinates to tracking user's eyes. If user's head or face move a lot, it takes time for the WebGazer recalculate the coordinate and the relative distance between the eyes and face. That would cause time delay or even failure tracking.

In addition, the background light matters. If the background light is dim, it means much noises in the background and it is difficult to for the camera to detect user' eye movement. Also, in a dim background, people's eyes would become gray or even black, that is a dead color for camera to catch the eye movement. And thus the total accuracy would decrease.

6.2 Advantages of Evaluation

we both did unit evaluation and total process evaluation. Unit evaluation is to evaluate the unit performance, function and accuracy. Total process evaluation is to evaluate the total performance, function and accuracy.

Unit evaluation is to evaluate the unit part of our application. We determine to evaluate each unit part once we finish that part in order to find error or malfunction early. Then we can clearly know what's the problem and how to improve it. If we don't do the unit evaluation, then our work will be a total mess. Unit evaluation can make sure our progress and finish our work on time.

Total process evaluation is to evaluate the total performance, function and accuracy finally. Even though the unit can make sure the unit part works well, it cannot make sure the total process also works well. Because the system performance is not just the sum of the unit performance. Thus at the last step, we did the total process evaluation to evaluate the total performance, function and accuracy. And we found some part that can be improved to have better performance.

6.3 Disadvantages of Evaluation

During our evaluation, we did unit evaluation and total process evaluation. However, we didn't find a professional tool or framework to follow to do the evaluation. So that means we may miss some part of the evaluation. We just roughly did unit evaluation and total process, however, we cannot make sure the unit evaluation is sufficient and effective. We need some library or some library or software tools to implement unit evaluation and total process evaluation.

7. Future Work

7.1 Expand Strengths

Our strengths are that we combine eye tracking with sketch recognition together to help user select items with just eyes. It has several directions to extend our strengths.

The first direction is to expand our selection system by adding more items, more categories, and more object moving paths. Currently, in each UI we only have at most 4 moving objects. And these objects are moving following the same path but at different locations all the time. In theory we can have more items in the screen and each item can move following specific path, like line, circle, square, triangle and so on.

The second direction is to expand the function of our application. Currently our application can only play movie, display books. Even though we have designed the interface for selecting food and change air condition temperature, we don't have hardware support. In future, we may add more functions like

play music, connect to online shopping website like Amazon to buy something, or even connect to fast food restaurant like Mcdonald's to order food directly. Also, if we get some funding, we may buy some related hardware to have hardware response to our application, like an air condition.

The third direction is to modify our application for business purpose. For instance, many customers stand in a line in the Mcdonald's talking with a cashier to buy hamburgers. It is time wasting and resource waste. What if there is a big screen in Mcdonald's and a customer standing in front of that screen with many hamburgers flying in that screen. Users just need to track the hamburgers with their eyes, the system will know what they want. It is efficient and effective for this purchase.

7.2 Improve Weaknesses

Our weaknesses are that our system can only recognize one user's eye tracking. In real business, we need to handle multiple users eye tracking problem.

8. Conclusion

8.1 Overview

A communication assistant application for paralyzed people is studied and evaluated. The application is based on normal webcam and totally written in JavaScript, which make this application available for every web user. The WebGazer.js API is applied for data collection. Collected eye-gazing data is processed and recognized to locate the object user want to choose, while using Rubine features and template matching algorithm. This application could help user to achieve certain purpose and the recognition accuracy for the application is about 60% during the evaluation

8.2 Highlights

Our project aimed at assisting paralyzed people to navigate through various pages easily using the eye-tracking interface to determine their interests in an easy fashion. Providing this all application as a Software as a service application proves this application to be easy to use from anywhere.

Following are some of the strengths and weaknesses of our project:

Strengths:

1) Our application can satisfy user's need to select items by eyes.
2) The application is a website, so if we push it online, anyone can use it with their own device.
3) The application runs entirely in the client browser, so no video data needs to be sent to a server.

Weaknesses:

1) The recognition is limited by the resolution of webcamera and the algorithm behind eye tracker API.

References

[1] Forster, K. I., & Forster, J. C. (2003). DMDX: A Windows display program with millisecond accuracy. Behavior Research Methods, Instruments, & Computers, 35(1), 116–124. doi:10.3758/BF03195503

[2] Mathôt, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. Behavior Research Methods, 44(2), 314– 324. doi:10.3758/s13428-011-0168-7

[3] Peirce, J. W. (2007). PsychoPy—Psychophysics software in Python. Journal of Neuroscience Methods, 162(1-2), 8–13. doi:10.1016/j.jneumeth.2006.11.017

[4] Schneider, W. (1988). Micro Experimental Laboratory: An integrated system for IBM PC compatibles. Behavior Research Methods, Instruments, & Computers, 20(2), 206–217. doi:10.3758/BF03203833

[5] Stahl, C. (2006). Software for Generating Psychological Experiments. Experimental Psychology (formerly "Zeitschrift für Experimentelle Psychologie"), 53(3), 218–232. doi:10.1027/1618-3169.53.3.218

[6] Adil Hamid Malla; Rahul Ashok Bhagat; Tracy Hammond. DyGazePass: A Gaze Gesture-Based Dynamic Authentication System to Counter Shoulder Surfing Attacks. IEEE International Conference on Identity, Security and Behavior Analysis (ISBA). January 10–12, 2018 | Singapore

[7] Dalmaijer, E.S., Mathôt, S., & Van der Stigchel, S. (2014). PyGaze: an open-source, cross-platform toolbox for minimal-effort programming of eye tracking experiments. Behavior Research Methods, 46, 913-921. doi:10.3758/s13428-013-0422-2