Webcam Based Eye Gaze Prediction System with Automatic Calibration for Web Browser

Niphat Karngumpol
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
n.karngumpol@gmail.com

Worapoj Kreesuradej
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
worapoj@it.kmitl.ac.th

***Abstract*—Now a day, researches about Eye Gaze Prediction Systems with General Webcam was interested by many researchers and has been continuously developed because it is cheap and can be applied in many ways. Currently, a Browser based open source library for eye gaze prediction has been developed, which is easy to use and can be developed in a variety fields. However, this method is still limited because of low precision and inconveniences because users must perform calibration every time before use it. Therefore, we propose ways to develop and solve the above problems. This research has two objectives. The first is to improve the accuracy of an existing Eye Gaze Prediction System. The solution is using the Simple Moving Average to reduce the volatile of results and increase accuracy. From a results, this method gives a test scores higher than the existing method with statistically significant at 0.01, calculated as 14.96 percent increase from average score of the old method. The second objective is to propose a solution for making the system can recalibrate itself to improve accuracy in a long time without having to perform calibration by the users. We record the gaze data from system in the first 1 minute and then take the recorded data to calculate the boundary of the screen. And then, compare the calculated boundary with the real screen boundary to find error factors. This calculated error factor has been applied back to the next result to increase accuracy. From the test results, we found that over time the average test scores gradually increase. And at the last minute, the average score from this method is higher than the average score from traditional method up to 13.66 percent.**

Keywords—eye gaze prediction; moving avarage; eye tracking; SMA; self-calibration;

I. INTRODUCTION

At present, research about eye-gaze prediction system has received attention and has been developed in variety fields of research. It has already been applied in various circles, including helping disabled people, preventing accidents of motor vehicles, advertising work, etc.

However, there are still many limitations for research at this time, for example, special equipment required with additional costs or needing to adjust some values, which makes it difficult to use. For budget-saving methods, results are still unstable and not accurate enough to be used in real situations.

From the above limitations, we want to develop a webcam-based eye-gaze prediction system that has the ability to fine-tune the accuracy by itself to create a new user interface and user experience that is easier for users and to improve precision and accuracy from existing systems

This research focuses on the development of eye-gaze prediction systems that work on the browser primarily because web technology and web-based applications are becoming increasingly popular. Therefore, the development of a browser-based system will help to apply this research in many different ways.

II. RELATED WORKS

In this research consists of related theories that are the basic concepts research as follows.

A. Eye-gaze prediction system

Eye-gaze prediction system [1][2] is a system that answers which point on the screen the user is looking at. The input is the image of the user's face and output is the coordinates on the screen that the user's eyes are looking at. Currently, Eye-gaze prediction system has been developed in many ways. The most popular method is detection with images from a webcam which can be divided into two main methods: infrared-based method, using infrared light and high resolution camera [3][4] and webcam-based method, using a general webcam [5][6].

In this research, we focus on a webcam-based method that can work on a web browser because we aim to develop a low-cost system that can be applied in a variety ways in real situations. At present, the webcam-based eye-gaze prediction system that can work on a web browser has already been developed called webGazer.js

B. WebGazer.js

In 2016, Alexandra and team propose WebGazer.js [7], the open source library for eye-gaze prediction developed for working with browser-based application. This library is developed with Javascript language and can work with general webcam. The important mechanism is Facial Landmark Detection, image-processing method to identify the position of the face and eyes. Then, paired a set of eye position with a set of screen position and train it into a prediction model. And then, take regression analysis to predict the current coordinates that the eye of the user is looking at.

C. Facial Landmark Detection

A method to search and identify a person's face from an images or videos. That is one of the important steps of eye-gaze prediction system. This method results in a set of coordinates of various important points of the face, allowing us to identify important positions on the face such as eyes, nose, mouth, cheeks, etc.

Currently, the browser-based open source library for facial landmark detection has been developed and used efficiently, including Clmtrackr[8], Track.js[9], BeyondFaceReality etc.

D. Regression analysis

Regression analysis [10] is a statistical method that studies the relationship of known variables called independent variation or Prediction Variable (X) which can be used to predict the value of another variable are called dependent variables. (Dependent variation) (Y). There are many regression methods such as linear regression, multiple regression, logistic regression, ridge regression, etc.

E. Multiple regression

Multiple regression [11], One of the regression analysis methods, which aims to create predictive equations, criteria variables with predictive variables by finding the relationship between variables in the form of predictions. This method consists of 2 types of variables: X is an independent variable or a predictor variable with 1 or more and Y is a dependent variable or criterion variable with 1

F. Simple Moving Average [12]

Moving Average is the calculation to find average value of a small group of data in the data series to make the data smooth, reduce volatility and make it easier to analyze. Generally used for stock and financial data analysis

III. IMPROVING THE EXISTING EYE-GAZE PREDICTION SYSTEM

In this research, there are two main steps to divide the experiment. At this step, we have adopted the existing eye-gaze prediction system to improve the results. We chose webgazer.js as a reference in the research because it is a system that has research paper supports and it also has a variety of practical applications.

We studied the problems of this existing system and found that one important obstacle that makes results from distortion prediction is inaccuracy of results. The results are very volatile. We hypothesize that if we can reduce this inaccuracy, the system will be able to provide more accurate results.

The solution we used to solve this problem is to use the Simple Moving Average (SMA) to make the results smooth.

A. Experiment setup

In the experiment, we used a 4m pixel camera on a Microsoft Surface laptop with a 13 inches screen size and 1420x840 pixel resolutions (resolution for web display). We chose the Javascript language to develop the system. And test the results on the Google Chrome browser.

B. Smoothing the results from the prediction

In principle, the Webgazer module will receive video images from a webcam and then process it frame-by-frame and provide predictive results in each time in the form of coordinates (x, y) through the variable x and y. We adopted the existing system by created a filter module to filter the results from the x and y variables to be more smooth by sending x and y into the filter. This filter is a queue that stores the values of x and y in any period of time and serves to find the average of x and y in that moment. When a new value is pushed into the queue, the oldest value will be pop out.

We can also customize a *queue_size* variable, which is the number of data in period of time that is calculated to find an average value.

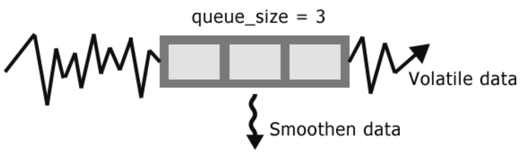

Fig. 1. SMA filter concept diagram.

C. Selecting queue_size

We notice that if the values of queue_size we define in the filter get higher, the accuracy will be increased. But the sensitivity of the result will be decreased or more delayed.

We make a proof by creating a precision and delay test set as following steps.

1) Let users to look at various points on the screen for a while and then save the current position of x, y and queue size from the prediction, stored it as an array. We change the queue_size value continuously from 1-30 and then take the recorded results in each period to calculate S.D. We record the average of every 5 S.D. in each queue_size value.

2) Test the system delay with Webgazer capability about calibration by mouse. When we move the mouse cursor, the result will also moves to the cursor point because the system considers it is a calibration.

We designed the experiment by moving the mouse to the specified point and record the timestamp that the mouse reaches that point, and then wait for the results from the prediction to move to the specified point and record the timestamp, and then find the difference of 2 timestamps. We repeat every 5 times in each queue_size value since queue_size = 1 to 30 and find the average value of that 5 times.

The results of the test can be shown as a Fig.2

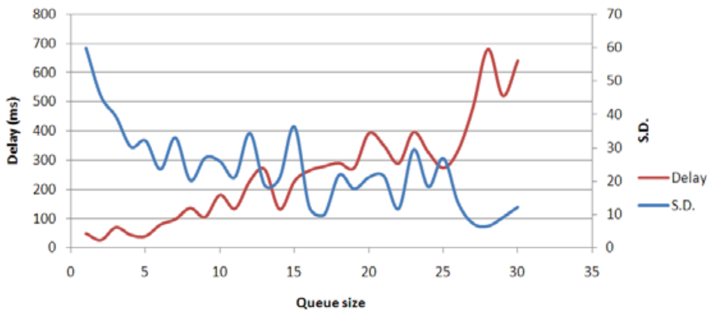

Fig. 2. A relation between delay rate, volatile rate (S.D.) and queue_size.

From the results, it can be seen that queue_size is inversely proportional to S.D. and varies with delay. When queue_size increases, the S.D. value decreases or the accuracy increases, but the delay will increase as well. From this test, we consider using queue_size = 17 because this value is in the area that is the intersection of the graph.

D. Experiment Result

After the existing eye-gaze prediction system was adopted by above procedure, we designed the measurement by let the volunteers to look at the ball moving randomly on the screen for 5 minutes per method. The system recorded the score from the gaze coordinates compared to the position of the ball every 500 millisecond. If the gaze coordinate point is far from the position of the ball more than 300 pixel, the score is 0. If the gaze coordinate point matches the position of the ball, the score is 100 points.

In the selection of volunteers, we use 10 volunteers who do not wear glasses with normal eyes shape or not too small eyes. The experiment was done in a room with a brightness of about 20-50 lux, where the light source was in front of users.

We defined *method A*: an old existing system, *method B*: a system we adopted in this step. The results from the test can be shown as Fig.3.

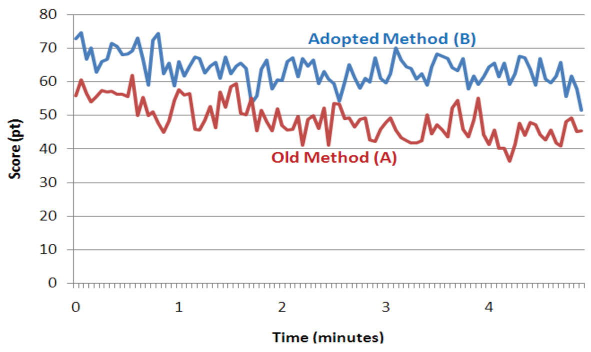

Fig. 3. The test scores from method A and method B in 5 minutes.

From the results found that when the data is smoothed, the results have a clearly higher score. However, the values of both methods tend to decrease as time increases.

IV. DESIGN AND DEVELOP A SELF-CALIBRATION SYSTEM

In the previous step, we found that when the time increased, the accuracy measured from the eye-gaze prediction system would be reduced. It may be because moving the position of the face too much causes a condition that does not match the calibration value at first time, resulting in an error. In addition, the existing system is necessary to do a new calibration every time to use the system, which may create inconvenience in the real application in many situations.

In this step, we designed and developed the automatic calibration feature to let the system to improve accuracy by itself. The principle is skipping the manual calibration process, but instead use the training data provided by ignoring the low-accuracy in the first minute, and then collect the gazing information, analyze it every 1 minute. In the analyzing, we find the boundary of the gaze-points that is expected to be points on the screen and compare it with the size of the real screen and then calculate the error factor and apply it back to results.

A. Step 1: Find the boundary of screen gaze.

Let users to look at the different position on the screen with scattered content for a while and then collect a gazing information every 100 ms for 1 minute.

The concept is: any area with the high density of the gaze-points is possible to be a screen area. Therefore, we can find the boundary of the point that is expected to be point on the screen by framing the area with high density of gaze-points.

B. Step 2: Calculate adjustment factor

If the system works correctly, the boundary obtained in step 1 should have a scale that is close to the screen's scale. At each time of processing, the system will adjust the tolerances by using 2 variables for each axis, i.e. magnitude and shifter.

magnitude_x = screen width / boundary width

*shifter_x = 0- (boundary_min_x * magnitude_x)*

magnitude_y = screen height / boundary height

*shifter_y = 0- (boundary_min_y * magnitude_y)*

C. Step 3: Apply the adjustment factor

In this step, we bring the magnitude and shifter from step 2 to apply to the prediction results.

*gaze_point_x = (gaze_point_x * magnitude_x) + shifter_x*

*gaze_point_y = (gaze_point_y * magnitude_y) + shifter_y*

The prediction system will be adjusted in every 1 minute by magnitude and shifter calculated in the new cycle will be calculated together with the previous cycle before use.

magnitude [t1] = magnitude [t0] x magnitude [t1]

shifter [t1] = (shifter [t0] x magnitude [t1]) + shifter [t1]

D. Results of the experiment

We use the same measurement method as the previous section, which let the volunteers to look at the moving ball and collect the prediction results. We defined a system in this step as *method C*. The test results can be shown as the Fig.4.

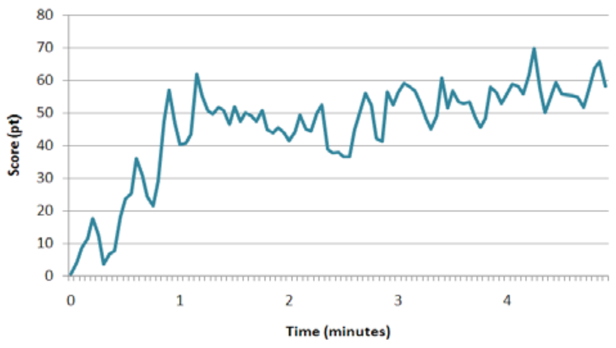

Fig. 4. The test scores from method C.

From Fig.4, the average score of all 10 people in the first time has a very low score. But over time, the average score gradually increased to a level close to the previous method and not likely to decrease.

V. Conclusion

From section 3, we improved the accuracy of the eye-gaze prediction system by creating a filter in order to smooth the results of the prediction. This filter can decrease a volatile of data and improve more accurate by applying the Simple Moving Average (SMA) concept. We choose 17 as a number of data in a period, which is a balance between accuracy and delay

The experiment was tested with 10 volunteers. The test was divided into 2 cases: *Case 1:* tested with an old existing system (method A). *Case 2:* tested with a system that is adopted according to the concept of this research (method B).

Based on the results, we found the method that obtained the highest average score in the 5 minutes was method B with an average score: 63.79 pt. which is higher than the method A with an average score: 48.83 pt.

We choose Paired-Sample T-Test method to confirm the hypothesis by assuming that.

- H_0: $u_2 <= u_1$
- H_a: $u_2 > u_1$

Determine alpha = 0.01

TABLE I. Paired sample T-Test result of both 2 methods

Scores	**N**	**Mean**	**S.D.**	**T**	**DF**	**Sig.**
Method A	99	48.83	5.47	23.7986	98	0.0001
Method B	99	63.79	4.44			

Based on the T-test analysis, T-state (23.7986) > T-critical (2.364) and p = 0.0001. Due to p <alpha (0.01), therefore reject H_0.

We can conclude that the adopted system in section 3 (method B) gives an average score higher than the old existing system (method A) with statistically significant at 0.01 or at 99% confidence rate, calculated as 14.96% higher.

However, we can observe that when time increases, the average score obtained in each period decreases in all cases. For this reason, we propose a solution to develop a feature for automatic calibrating by itself.

We import training dataset into the system before starting to use. When the system starts, users do not have to manually calibrate every time, like the old system. We record the gazing information from the system every 1 minute and find the area that is likely to be the edge of the screen by considered a high-density gaze point. And then calculate the error factor by comparing the boundary obtained from the calculation with the actual boundary of the screen. Then take the error factor to apply to the system in the next round.

Based on the results, we found that the average score of everyone in the first minute of the test was very low, but began to rise after passing the first minute onwards and rising until the score range was close to the previous method.

If consider at the last minute of the test. We found that the average score of method C in section 4 increased to 57.65 pt., which calculated as 13.66% higher than the score of method A, which falls to 43.99 pt., without the need for calibration by user themselves before use.

However, this research also has obstacles that need to be developed to solve it such as an accuracy was dropped when faced with low light conditions, problems for users who wearing eye-glasses, problems caused by users looking at a some point on a screen for too long which results on the error of calculating the possible screen boundary.

In addition, this research needs to be developed for use in real situations. There are many concepts that can be developed from this research, such as the analysis of reading behavior on the screen, creating eye-command system for paralysis patient in low price, etc. These will help to create new technology choices for users in every position.

References

[1] Yunlong Feng, Gene Cheung, Wai-tian Tan, Patrick Le Callet, and Yusheng Ji, "Low-Cost Eye Gaze Prediction System forInteractive Networked Video Streaming" IEEE Transactions on Multimedia 15(8), pp. 2-4, Jul 2012.

[2] M. Reale, T. Hung, and L. Yin, "Pointing with the eyes: Gaze estimation using a static/active camera system and 3D iris disk model," in IEEE International Conference on Multimedia and Expo, Singapore, July 2010.

[3] Bin Li, Hong Fu 3,Desheng Wen and WaiLun LO, "Etracker: A Mobile Gaze-Tracking System with Near-Eye Display Based on a Combined Gaze-Tracking Algorithm". Sensors 2018, 18, 1626, May 2018.

[4] Ehsan Arbabi, Mohammad Shabani and Ali Yarigholi, "A low cost non-wearable gaze detection system based on infrared image processing", Available online: 2017.

[5] Atul Sahay, Pradipta Biswas, "Webcam Based Eye Gaze Tracking Using a Landmark Detector", Compute '17 Proceedings of the 10th Annual ACM India Compute Conference, p31-37, Nov 2017.

[6] Pingmei Xu Krista A Ehinger? Yinda Zhang Adam Finkelstein Sanjeev R. Kulkarni Jianxiong Xiao, "TurkerGaze: Crowdsourcing Saliency with Webcam based Eye Tracking" arXiv:1504.06755v2 [cs.CV] 20 May 2015.

[7] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, James Hays, "WebGazer: Scalable Webcam Eye Tracking Using User Interactions", NSF grants IIS-1464061, IIS1552663, 2016.

[8] Audun Mathias, "clmtrackr: Javascript library for precise tracking of facial features via Constrained Local Models.", https://github.com/auduno/clmtrackr, 2014, [Online; accessed 2015-07-08].

[9] Eduardo Lundgren, Thiago Rocha, Zeno Rocha, Pablo Carvalho, and Maira Bello, "tracking.js: A modern approach for Computer Vision on the web", http://trackingjs.com/, 2014. [Online; accessed 2015-05-15].

[10] Marko Sarstedt, Erik Mooi, "Regression Analysis", A Concise Guide to Market Research (pp.193-233)

[11] Catherine Connolly, "The Use of Multiple Regression Analysis in Employment Discrimination Cases", 10 Population Res. & Pol'y Rev. 117 (1991).

[12] Puchong Praekhaow, "Determination of Trading Points using the Moving Average Methods", GMSTEC 2010: International Conference for a Sustainable Greater Mekong Subregion, 26-27 August 2010.