ROBUST EYE-GAZE TRACKING AND APPLICATION

Jiachen Tian
Swanson School of Engineering
University of Pittsburgh
Pittsburgh, USA
tjc990806@gmail.com

***Abstract*— Eye-gaze tracking can be widely implemented in numerous areas; Popular algorithms include Hough transform, KCF tracker, Optical flow, and convoluted neural network. However, it still poses a challenge on tracking eye-gaze under noisy conditions or when the pupil is barely distinguishable from the background. This research would introduce practical ways to track eye-gaze under nonideal conditions when the inputs are a series of MRI videos with noises covering the frames. The filters, preprocessing, and machine learning model implemented would also be covered in-depth to analyze its success in tracking the eye-gaze position, the resulting boost in speed and immutability to constantly changing noises in the video.**

Keywords—Estimation of eye gaze; machine learning; image processing; gear free

I. INTRODUCTION

Infrared Eye Tracking is an eye-tracking setup free of any headset or extra hardware setup apart from having an infrared camera placed next to a display. The infrared light aims at the eyes, which could be detected by the camera but will not disturb the users[1]. The reflection and the pupil's center displacement would be tracked and analyzed for the user's precise gaze-position. The computer would record the infrared camera data as avi video files and store them for further analysis due to the integrated eye-tracker's sensitivity to noise. The task consists of five repeated events instructing participants to make expected saccades: (i) Object gazing at the center of the screen on cue; (ii) Object gazing at the image when an image appeared in one of four non-central locations; (iii) Object gazing back at the center on cue; (iv) Object gazing to where he or she remembered the image was; (v) Object gazing back at the center of the screen on cue; to five established gaze positions. Each of these stages (A thorough demonstration of each stage is shown in Figure 1) can be measured against an expected value to provide a score of each participant's behavior.

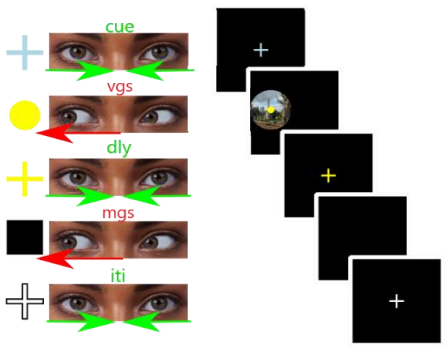


Fig. 1. Demonstration of five instructions

A. *Problems to tackle and potential solutions*

Proprietary solutions are provided by most hardware vendors but are not interchangeable and are difficult or impossible to extend. With the existing computer vision algorithms, gaze tracking could readily be achieved. However, challenges remain to precisely and promptly track eye-gaze directions under noisy conditions and when the objects are hardly distinguishable from the background. Apart from tracking, the asynchronous timing between datasets and frames would also pose a big challenge. The process involves four distinct parts. (i) Filtering: Filter out any noise and outliers in the frames to only grab useful pixels for later analysis. Methods include Gaussian blur, threshold, and canny image. (ii) Pupil Tracking: Frame to frame tracking and precise evaluation of pupil positions. Methods include Hough Transform and KCF tracker. (iii) Gaze direction analysis: Sync the datasets given and the datasets retrieved before analyzing the distance change among all the different stages.

II. VIDEO FILE PREPROCESSING IN DETAILS

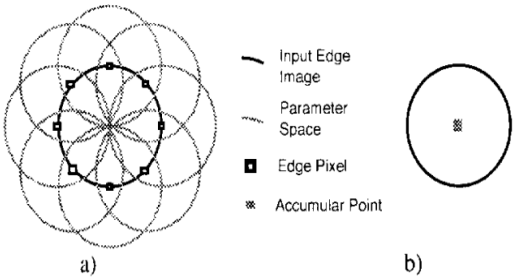
Hough transform is a powerful computer vision tool to detect lines, circles, ovals, or any other polygons in an image. We focus on circle detections as part of the research goal. Hough circle transform's main idea is to take three points that define a circle in a 2D world(x, y, r). Which could be described as [2, eq.(1)]

$$(x-a)^2+(y-b)^2=r^2 \tag{1}$$

where (a,b) is the center of the circle, and r is the radius. By transforming the circle from a standard coordinate system to a polar coordinate system, points originally in the peripheral could be represented as central locations of the circle[3, eq(2,3)]

$$x = a + rcos(\theta) \tag{2}$$
$$y = b + rsin(\theta) \tag{3}$$

as shown in [3, Fig. 2]. To decide which location forms a circle, an r value will be randomly defined, iterating theta for 360 degrees and changing x and y for each iteration. If multiple points in the 2D standard coordinate system correspond to one point in the 3D polar coordinate system, then a central point for the circle with the radius of r has been located and later implemented. However, due to its voting mechanism, the Hough Transform algorithm could still be affected by noises which are prevalent in our video datasets. Therefore to boost up Hough Transform result's accuracy, preprocessing steps are unavoidable, which includes image blurring, image thresholding, and canny image transformation.



a) The contribution of edge points to the accumulator space, b) Edge point contribution to a single accumulator point [3].

Fig. 2. Hough Transform Detection

A. *Steps of Preprocessing*

As shown in Figure 3, noises covering the whole frame, which constantly changes for different time frames, makes it hard for the algorithm to determine the pupil's and glint's precise location. Therefore to reduce noise and to generate more reliable frames for testing, Gaussian blur is implemented to blur the frames extracted from video file and to remove any unnecessary noise in the frame. Gaussian blur can be represented by [3, eq.(4)].

$$G(x,y) = \left(\frac{1}{2\pi\sigma^2}\right) e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad (4)$$

A convolution matrix is built with the formula, which applies to the original image. The level of preservation using gaussian filters would highly benefit the tracking of glint due to its smaller size. Even though the blurring factor can be kept identical across all the frames in one video file, it also depends on image size, illumination, noise patterns, and other factors that may vary across different video files. Therefore, in future improvements, the blurring factors will be automatically generated using machine learning and statistical analysis for different video files to get the best outcomes [3].

The threshold of an image is straightforward. If the pixel value is greater than a threshold value defined, it will be assigned white. Otherwise, it will be given black. Like blurring, it needs two parameters (lower threshold, higher threshold), which varies depending on image size, illumination, noise patterns, and other distinct factors for every video file. Therefore, the same as blurring factors, machine learning, and statistical analysis will be implemented to retrieve the best threshold value for tracking performance's most accurate outcomes.

Canny edge detection is a technique to extract useful structural information from different visual objects. The edge consists of two directions: x and y. However, canny factors are not as crucial as blurring factors or threshold factors. As long as the threshold image is optimized based upon the pre-defined canny parameters, it would result in a reasonable canny image for successful Hough Transform [4]. Therefore, to optimize the runtime efficiency, canny factors would be a constant value defined by users

.

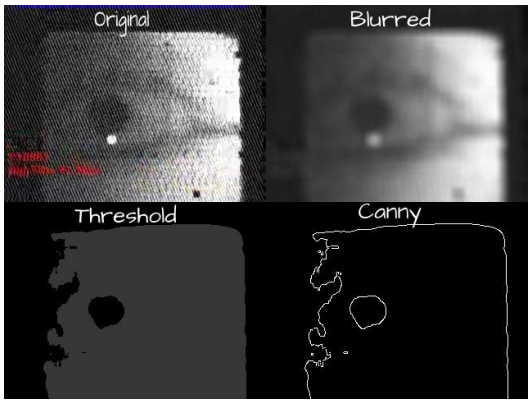


Fig. 3. Three stages filtering with constant pre-defined parameters

III. Data Preprocessing and Detailed Tracking

As mentioned, due to the variations in image size, illumination, and noise patterns, the whole process needs to undergo a series of data preprocessing steps to get the optimized results for blurring factors and thresholding factors. To first retrieve the thresholding factors, the tracker would be implemented on the number of frames determined by users by iterating on different threshold factors with the same blurring factors and same canny factors. When other factors are set constant, the trail that forms the largest numbers of circles would be considered the best. For which we could take advantage of the voting mechanism of the Hough Transform. Suppose a trial has the most votes and the detected pupil in each frame for a test is within certain restrictions. In that case, variations in size and position with glint, the threshold parameters correspond with it would be stored as the best parameters shown in Figure 4.

Before retrieving the optimized threshold factors, the blurring factors should be determined first because it could vastly reduce the chance of unnecessary edges forming different circles. The blurring factors could be determined by scanning through a certain number of frames with blurring factors and threshold factors continually changing within a specific range defined by users. Ideal data should be repeated, and the differences between frames should not be excessive due to pupil motion nature. The best blurring factors could be retrieved by finding the pre-testing trail with the smallest standard deviation, as shown in Figure 5.

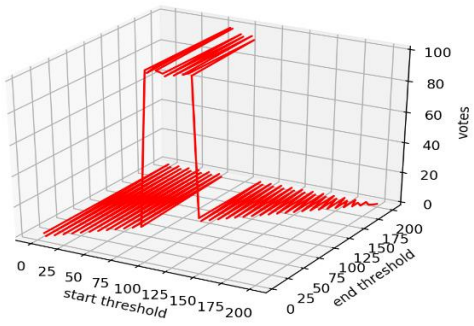


Fig. 4. Plot showing votes under ideal blurring factors and different thresholding factors

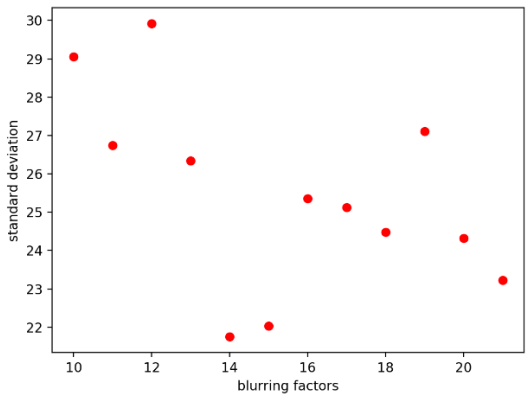


Fig. 5. Plot showing standard deviation under changing threshold factors blurring factors

A. *Preprocessing for glint*

The preprocessing steps for glint detection are nearly identical to pupil preprocessing. But blurring factors are unnecessary or could be small due to the size of the glint, which means it could easily be missed by Hough transform if we blur it further. In terms of threshold factors, an automatic threshold calculator is implemented. Compared with pupil, glint is more distinguishable from the background and easier for the corresponding algorithms to recognize. The Otsu Thresholding is implemented, which could automatically find the best threshold factors for the glint. Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels on each side of the threshold, i.e., the pixels that fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum [5]. Figure 6 shows the prioritized threshold result when applying Otsu's Threshold on the original image with the gaussian blur factor (1,1). However, prioritizing threshold factors will not give us the best result for glint detection. Unlike the pupil, glint is reflected in the infrared camera, which leaves it more vulnerable to noise. The more noise there is, the less probable that the glint would form a circle. Therefore, after getting the optimized threshold values and the corresponding canny image with canny value (40, 50), another preprocessing step should be implemented that calculates the standard deviation of the tracking results from the number of frames defined by the users by changing the voting factors needed from Hough Transform. Then, even if the glint does not resemble a circle's shape, the Hough transform would still vote it as a circle. To get the best result, the voting factor with the minimum standard deviation (because the data is repeated, the smaller the standard deviation, the better) would be selected to track the glint position.

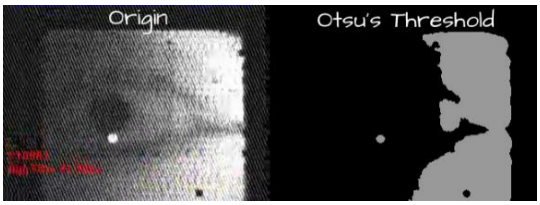


Fig. 6. Finding the glint using Otsu's Threshold

B. *KCF tracker as supplemental tracker*

Apart from implementing Hough Transform to precisely pinpoint pupil and glint in different frames, the KCF tracker is also implemented as a supplement when it comes to the cases Hough Transform does not quite work. Despite its incapability of tracking the precise location, the KCF tracker could, in turn, be used to filter out the wrong values (outliers) from the Hough Transform. Rectangular boxes selected by users will be used to track pupil and glint in the KCF tracker. Therefore, an interface is constructed to enable users to select the bounding box for both glint and pupil, as shown in Figure 7. Users can select a bounding box on the left of the interface, and the results would be dynamically drawn on the right. Despite the blinking, most frames in the video contain both pupil and glint. Therefore the KCF tracker is initialized on a fully opened-eye frame. By iterating and summing over pixel values in an image, could the best opened-eye frame be found since the bigger the pixel, the darker the image, and the pupil is considered the darkest.

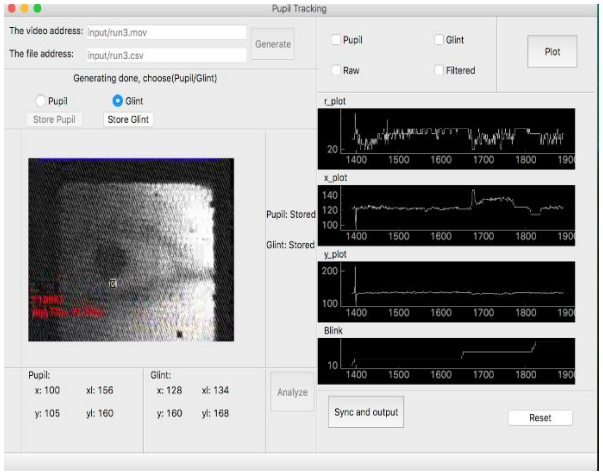


Fig. 7. User interface that enables selections and animation of tracking results

C. *Necessary help from user interface*

By letting the user select an area bounding either pupil or the glint, not only do we improve pre-processing and tracking performance to a large extent, but we also minimize the frequency of occurrence of outliers. It is also a crucial step in automatically determining the Otsu threshold of the glint because only by selecting areas specific to the glint could the algorithm aptly separate glint from the background, thus getting the best threshold values for the glint. After the optimal threshold is retrieved from the users' area, the area size will be incremented by specific values. All the glint movements in the video would be included. Therefore, the defect of scanning through only the user-defined area is that it could only be done once, and it will not be able to handle illumination in the later frames from the same video file. However, by applying some more filters, accurate results could still be retrieved.

IV. TRACKING RESULTS DEMO

After all the preprocessing and tracking are completed, the user interface generates three types of outcomes: (i) Tracking results plotted onto each frame for every frame inside the video as shown by Figure 8; (ii) Tracking results plotted for the whole video(x, y, r, and blink frequency) as shown by Figure 9 and Figure 10. To keep it clean, Figure 9 only demonstrates movement in the x-direction since that is the majority of trends in this specific case. Figure 10 shows the blinking rate of the subject throughout the experiments. (iii) Tracking results of x, y, r, and blinking rate output to multiple CSV files for later analysis. However, as shown from the x-

position plot of both glint and pupil in Figure 8, since the video's frame rate is 60f./s, with such intensity, the plot would look quite noisy even with all the filters applied to the frames before initialing the tracker. Therefore, further filtering of the resulting data is conducted to remove noises and to make the plots more readable.

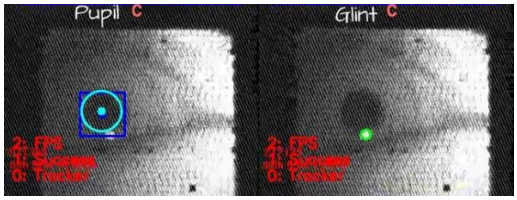


Fig. 8. Tracking results plotted onto original frame for both pupil and glint

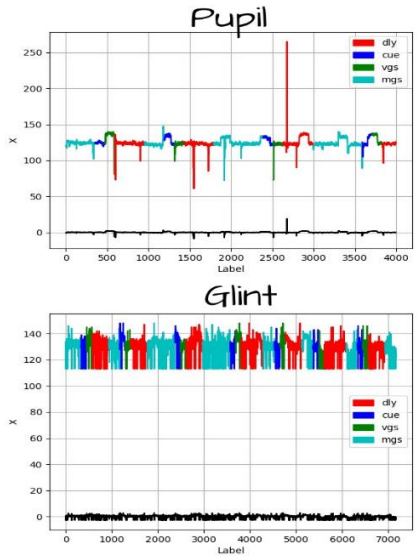


Fig. 9. Unfiltered position plot showing the motion in x direction. Black line underneath shows z-scores for the data

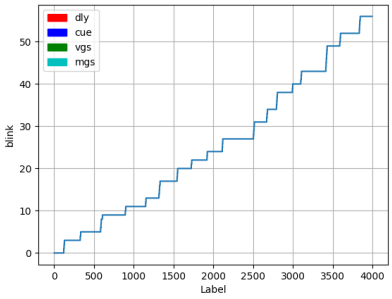


Fig. 10. Target Blinking rate plot that is not labelled by colors

A. *Result data filtering*

The first resort of filtering the resulting data is to compare the radius and position of glint and pupil so that the data make sense when putting them together, i.e., the radius of the glint should never be bigger than that of the pupil. After completing the first pass of the filter, we use a z-score to further filter the data and make the plot more human-readable. Z-score is calculated based upon the standard deviation of a range of data defined by the user (in this case, 2000). It measures exactly how many standard deviations above or below the mean of that chunk of data. The formula can be written as equation (5).

$$z = \frac{data\ point - mean}{standard\ deviation} \tag{5}$$

The more extensive the range of data, the more transparent the filter results are going to be. The results are shown in Figure 11 after the appropriate filter is imposed on the original data.

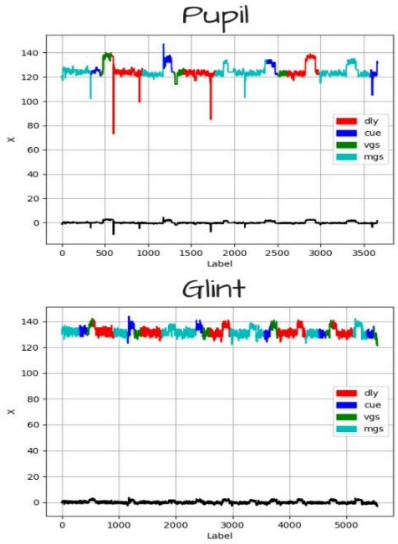


Fig. 11. Filtered position plot showing the motion in x direction. Black line underneath shows z-scores for the data

V. IMPLEMENTATION OF MACHINE LEARNING FOR RESULT ANALYSIS AND MATCHING

To get more intuitive results and the best performing modules for precision analysis, Random Forest (A very popular machine learning algorithm) will be implemented to train, evaluate, and generate the data module, as shown in [9, Fig. 12]. First off, a CSV file needs to be generated, which contains all the data from tracking outputs and the original data file. The resulting file includes outcomes - cue, vgs, dly, mgs, iti - as discrete variables and x, y, r as continuous variable inputs.

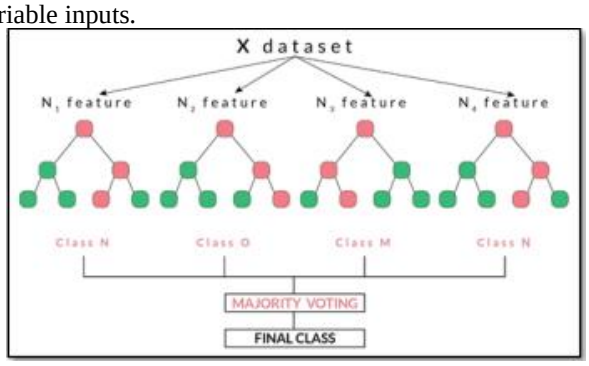


Fig. 12. Random Forest structure

A. *Generating and analyzing machine learning modules*

After successfully constructing the necessary file using the tracking results and the origin file data, the data is split into 75% training cases and 25% testing cases. TidyModels in r would be implemented to process and train the test cases and cross-validate the forest pruning data sets. After the Random Forest model's successful generation, the module is used to evaluate the testing cases. The results are shown in Figure 13, with a ROC value reaching 0.82, which is

considered very accurate in performance. To get a better understanding of the whole data, more metrics will be evaluated, including variable importance shown in Figure 14, which shows the weights of each variable.

	.metric	.estimator	.estimate	.config
	<chr>	<chr>	<dbl>	<chr>
1	accuracy	multiclass	0.624	Preprocessor1_Model1
2	roc_auc	hand_till	0.825	Preprocessor1_Model1

Fig. 13. Prediction accuracy

x	y	r
320.7913	161.4464	182.7877

Fig. 14. Variable importance

After all the data analysis and forest tuning are completed, a training module would be generated based upon x, y, and r to classify the result: cue, vgs, dly, mgs and iti. Currently the machine learning module is trained using only 2000 datasets that are manually labelled. It would later be integrated into the user interface and retrained using new data generated every time the app is called. As a result, it would be able to generate a universal machine learning module that is able to readily and swiftly handle all the corner cases and outliers normal statistical analysis failed to handle.

VI. Conclusion

In this study, we propose and evaluate a method for tracking eye-gaze on bad quality videos taken by MRI. The process focuses primarily on monitoring pupil and glint using Hough transform and KCF transform. Many preprocessing steps are conducted on both glint and pupil for precision and runtime efficiency. Finally, to further boost up the runtime efficiency and precision, a random forest model is constructed. As a result, it is confirmed that the program can precisely track the eye-gaze position.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] P. Doungmala and K. Klubsuwan, "Helmet Wearing Detection in Thailand Using Haar Like Feature and Circle Hough Transform on Image Processing," 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, 2016, pp. 611-614, doi: 10.1109/CIT.2016.87.

[3] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[4] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[5] M. A. B. Siddique, R. B. Arif and M. M. R. Khan, "Digital Image Segmentation in Matlab: A Brief Study on OTSU’s Image Thresholding," *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, Dhaka, Bangladesh, 2018, pp. 1-5, doi: 10.1109/CIET.2018.8660942.

[6] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[7] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[8] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[9] J. S. Park, H. Sung Cho, J. Sung Lee, K. I. Chung, J. M. Kim and D. J. Kim, "Forecasting Daily Stock Trends Using Random Forest Optimization," 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), 2019, pp. 1152-1155, doi: 10.1109/ICTC46691.2019.8939729.