ROBUST EYE-GAZE TRACKING APPLICATION

Jiachen Tian, jit29@pitt.edu

Abstract*—Eye gaze points could be measured in one of several ways, which falls into three big categories: (i) with contact lens attached to the eye; (ii) Computer vision tracker without direct contact to the eye; and (iii) electric potentials measurement by placing electrodes around the eyes. Popular algorithms include Hough transform, KCF tracker, Optical flow, and convoluted neural network. This paper would introduce the Hough Transform algorithm as the primary tracking algorithm which has proven its reliability when it comes to robust shape detections. The filters and preprocessing implemented would also be covered in-depth to analyze the resulting boost in speed and reliability after application.

KeyWords— Hough Transform, KCF tracker, Preprocessing, Optical flow, Convoluted Neural Network

INTRODUCTION: HARDWARE SETUPS AND DATA SETS

Infrared Eye Tracking setups

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. The reflection and the displacement of the pupil's center would be tracked and analyzed for the precise gaze-position of the use. Due to the integrated eye-tracker's sensitivity to noise, the computer would record the infrared camera data as avi video files and store them for further analysis.

Position Datasets

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-centeral 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.

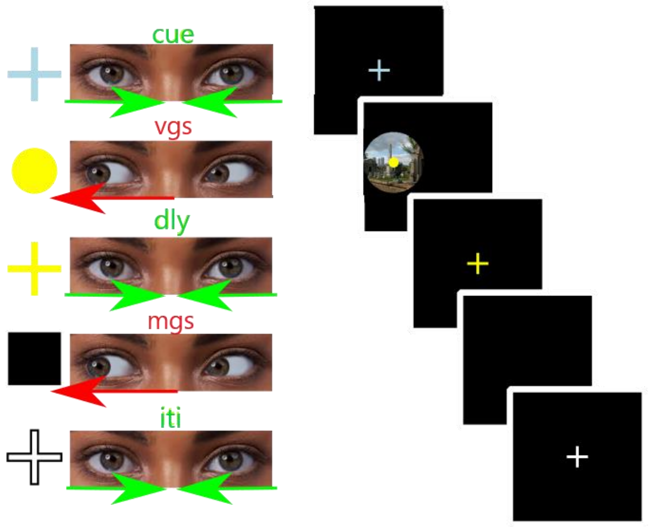


FIGURE1
Demonstration of the five instructions

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 are given and the datasets retrieved before analyzing the change of distance among all the different stages.

VIDEO FILE PREPROCESSING IN DETAILS

Hough Transform: A Secured Solution for pupil and glint detection

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: $(x-a)^2+(y-b)^2=r^2$, 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:$(x-a)^2+(y-b)^2=r^2$, as shown in Figure 2.[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.

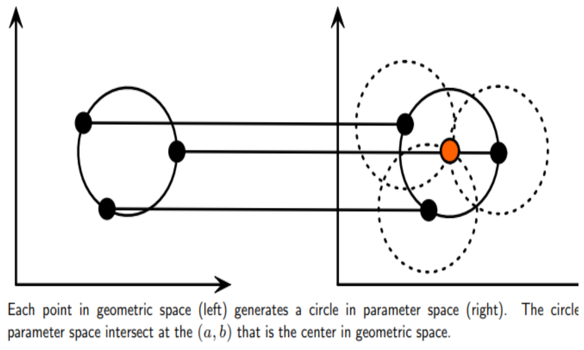

Each point in geometric space (left) generates a circle in parameter space (right). The circle
parameter space intersect at the (a, b) that is the center in geometric space.

FIGURE 2 [2]
Demonstration of circle in polar coordinate system

However, due to its voting mechanism, the Hough Transform algorithm is vulnerable to noises which is prevalent in our video datasets. Therefore to boost up the accuracy of Hough Transform results, preprocessing steps are unavoidable, which includes image blurring, image thresholding, and canny image transformation.

Blurring Factors

As shown in Figure 4, The unordered noise covering the whole frame 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.

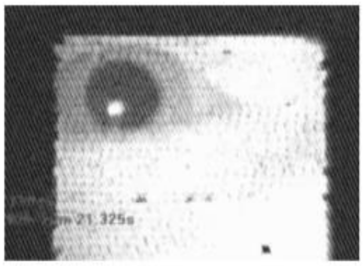


FIGURE 5

Original frame from a single video file

Gaussian blur can be represented by the formula $G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$. In the formula, x is the distance from the origin on the horizontal axis, y is the distance from the origin on the vertical axis, and sigma is the standard deviation of the gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values calculated using the formula shown above are used to build a convolution matrix, which applies to the original image. The heaviest weight will be received by the original pixel(having the highest Gaussian value), and the neighboring value receives smaller weights as their distance to the original pixel increases. The previous process results in a blur that preserves boundaries and edges better than other filters, which would benefit glint detection due to the glint's small size. [3]

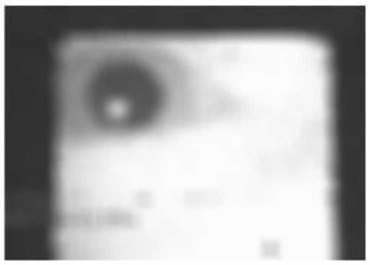


FIGURE 5
blurred image with a blurring factor of (120, 120)

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 outcome.

Threshold Factors

The threshold of an image is straightforward. If the pixel value is greater than a threshold value defined, it will be assigned a value (white in most cases). Otherwise it will be assigned another value (black in most cases). Like blurring, it needs two parameters (Lower_threshold, Higher_threshold), which varies depending on image size, illumination, noise patterns, and other factors which are distinct 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 the most accurate outcomes of tracking performance.

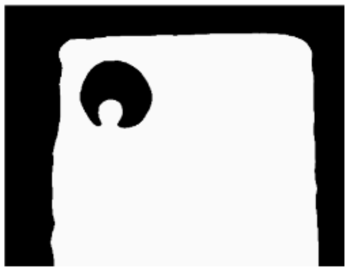


FIGURE 6
Threshold image with a threshold factor of (120, 250)

Canny Factors

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of processed data. There are five steps involved in detecting a canny image: (i) Find the intensity gradients of the image; (ii) Apply non-maximum suppression to get rid of spurious response to edge detection; (iii) Apply double threshold to determine potential edges; (iv) Track edge by hysteresis. The edge consists of two directions: x and y. However, canny factors are not as crucial as blurring factors or threshold factors because as long as the threshold image is optimized, any canny factor would result in a reasonable image for successful Hough Transform.[4]

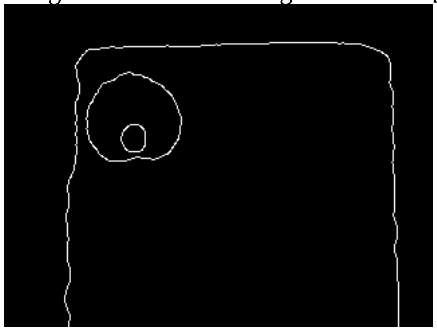


FIGURE 7
Canny image with canny factor of (40, 50)

DATA PREPROCESSING IN DETAILS

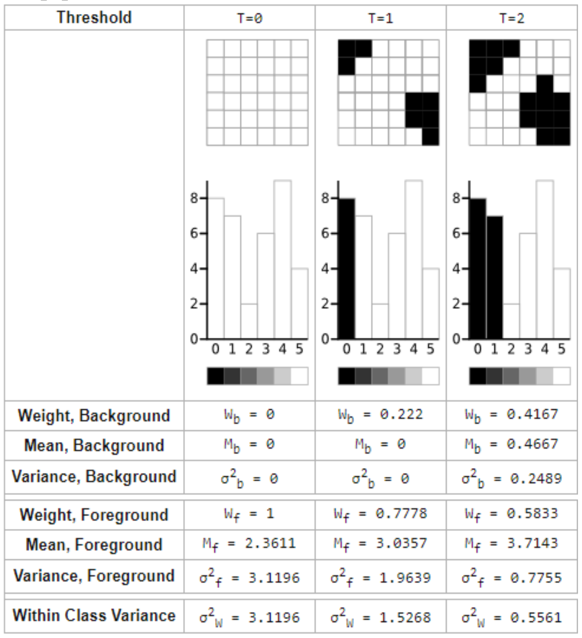
Get the optimized blurring factors and thresholding factors for successful image tracking

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 get 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 for the number of iterations defined by users. Then the best results would be merely taking the parameters of the most voted trails. Because threshold factors determine how well the canny image is, the quality of the canny image determines the quality of the circle, which Hough Transform detects and votes on.

After the threshold factors are successfully retrieved, the blurring factors could be determined based upon threshold factors by scanning through the tracker results using standard deviation. Since ideal data should be repeated and the differences between frames should not be excessive due to pupil motion nature. Therefore, the best blurring factors could be retrieved by finding the pre-testing trail with the smallest standard deviation.

Preprocessing for glint

The preprocessing steps for glint detection are nearly identical to pupil preprocessing. But blurring factors are unnecessary due to the small 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 since, compared with pupils, 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 pupil. 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]



Threshold	T=0	T=1	T=2
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$
Variance, Background	$\sigma^2_b = 0$	$\sigma^2_b = 0$	$\sigma^2_b = 0.2489$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$
Variance, Foreground	$\sigma^2_f = 3.1196$	$\sigma^2_f = 1.9639$	$\sigma^2_f = 0.7755$
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$

FIGURE 8 [5]
Processes that split foreground from background and calculating the minimum variances for different threshold factors

Figure 8 shows the comparison between different steps in calculating parameters that determine the best threshold factors, which would later be chosen as the threshold for glint detection. However, prioritizing threshold factors will not give us the best result for glint detection. Unlike pupil, glint reflects light from 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 results, another pre-processing 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 to decide how many votes would be to decide a successfully identified circle. 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.

KCF tracker as supplement 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 9. Users are able to select a bounding box on the left of the interface and the results would be dynamically drawn on the right.

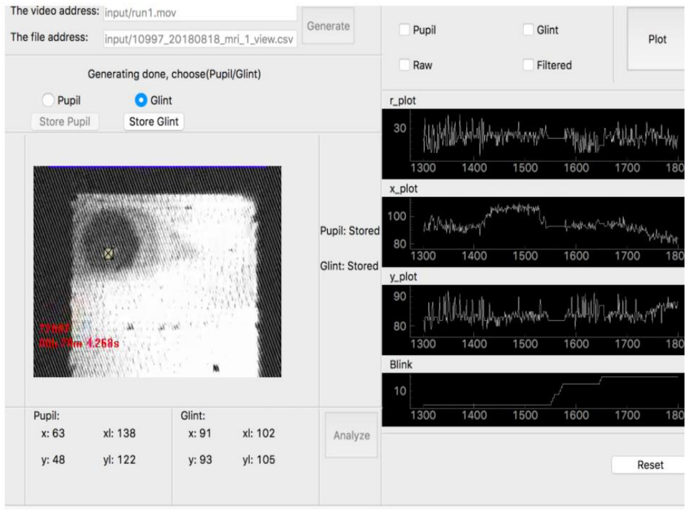

The video address: input/run1.mov
The file address: input/10997_20180818_mri_1_view.csv
Generating done, choose(Pupil/Glint)
Pupil [✓ Glint]
Pupil: Stored
Glint: Stored
Pupil:
x: 63 xl: 138
y: 48 yl: 122
Glint:
x: 91 xl: 102
y: 93 yl: 105

FIGURE 9
User Interface that enables selection and animations of tracking results

The basic idea of the correlation filter tracking(KCF) is estimating an optimal image filter such that the filtration with the input image produces the desired response. The desired response would appear as a gaussian shape centered at the target location. The filter is trained from translated(shifted) instances of the target path. When testing, the filter's response is evaluated, and the maximum gives the new position of the target. The filter is trained on-line and updated successively with every frame for the racker to adapt to moderate target changes. Some other available trackers can outperform the KCF tracker's accuracy, such as Boosting, CSRT, and TLD. However, KCF is chosen as the supplement tracker due to its computation efficiency. Its computation can be performed efficiently in the Fourier domain. Thus the tracker can run super-realtime while maintaining accuracy.[6]

Necessary help from user interface

By letting the user select an area bounding either pupil or the glint, not only 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 area selected by users, the area size will be incremented by specific values so that 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.

TRACKING RESULTS DEMO

After all the preprocessing and trackings 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 10; (ii) Tracking results plotted for the whole video(x, y, r, and blink frequency) as shown by Figure 11 and Figure 12. To keep it clean, Figure 11 only demonstrates movement in the x-direction since that is the majority of movements in this specific case. Figure 12 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 as shown by Figure 13.

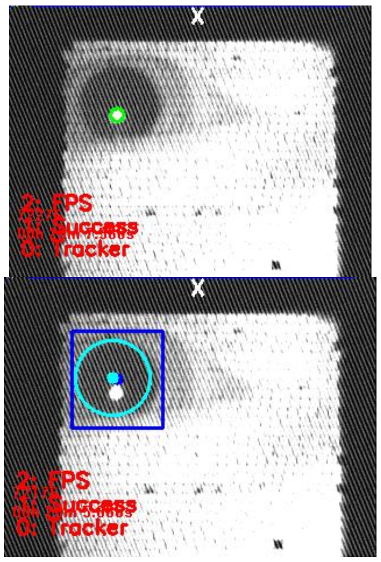


FIGURE 10
Tracking results plotted onto original frame for both pupil and glint

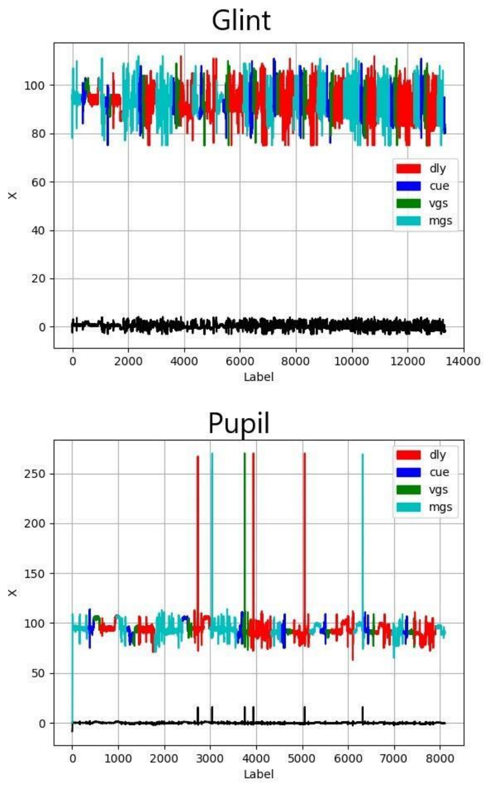


FIGURE 11
Unfiltered position plot showing the motion in x direction. Black line underneath shows z-scores for the data

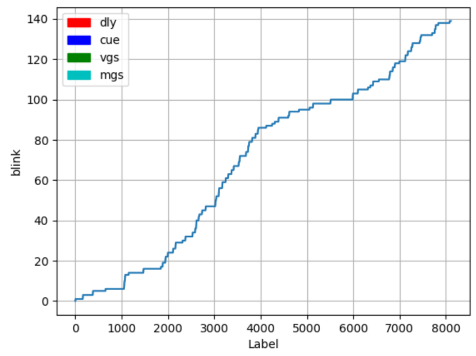
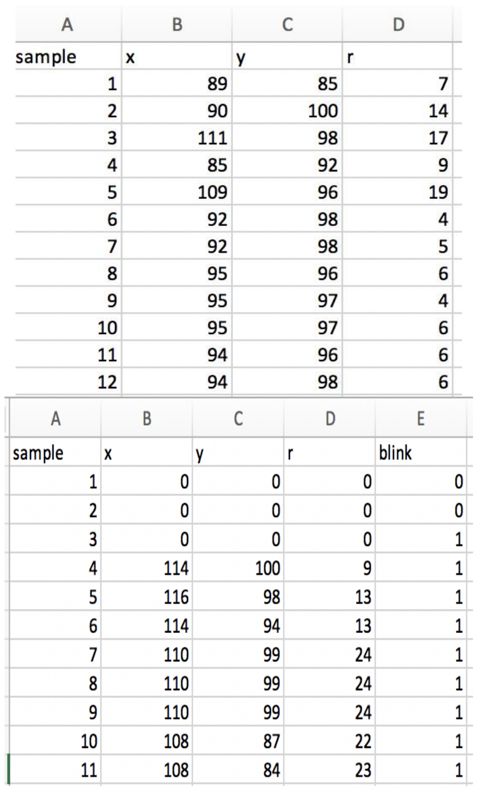


FIGURE 12
Target blinking rate plot that is not labeled by colors



A	B	C	D
sample	x	y	r
1	89	85	7
2	90	100	14
3	111	98	17
4	85	92	9
5	109	96	19
6	92	98	4
7	92	98	5
8	95	96	6
9	95	97	4
10	95	97	6
11	94	96	6
12	94	98	6

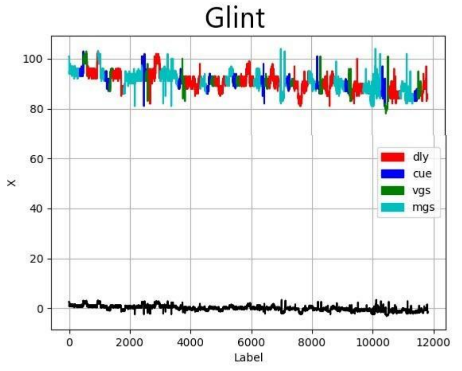
A	B	C	D	E
sample	x	y	r	blink
1	0	0	0	0
2	0	0	0	0
3	0	0	0	1
4	114	100	9	1
5	116	98	13	1
6	114	94	13	1
7	110	99	24	1
8	110	99	24	1
9	110	99	24	1
10	108	87	22	1
11	108	84	23	1

FIGURE 13
Data outputted to the csv file, glint(top), pupil(bottom)

However, as shown from the x-position plot of both glint and pupil in Figure 11, since the frame rate for the video 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 further remove noises and to make the plots more readable.

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 filter the data further 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 $z = \frac{data\ point - mean}{dtandard\ deviation}$. The more extensive the range of data, the more transparent the filter results are going to be. The results are shown in Figure 14 after the appropriate filter is imposed on the original data.



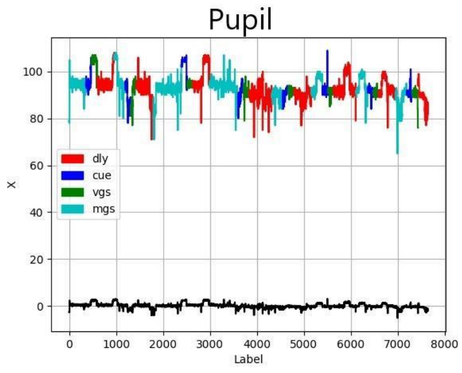


FIGURE 13
Filtered position plot showing the motion in x direction. Black line underneath shows z-scores for the data

As shown by the graphs, the motion in x direction for both glint and pupil is clearly plotted. To further rationalize the data, the plot is colored based on four stages mentioned at the beginning of this paper – cue, vgs, dly, and mgs. The ITI event is integrated 2 seconds after mgs since it's the same as cue. Likewise, ISI is merged into dly. Figure 14 shows a close up look on the coloring of the plot.

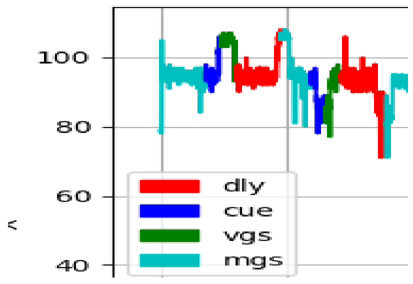


FIGURE 14
Zoomed plot showing the plot marked by colors representing five stages.

It is clear that there is a lag between the original dataset and the dataset generated by the tracker because the cue (Represented by blue) should be flat as it cues the user to look at the center. To resolve this issue, we could again use standard deviation, by iterating through every cue section, finding the smallest standard deviation, and then get the lag time. As shown in Figure 15, the plot looks more reasonable as we counteract the lag time between the original data and the video frames.

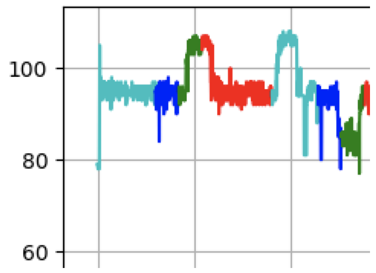
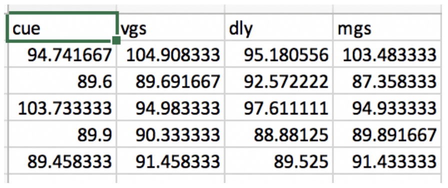
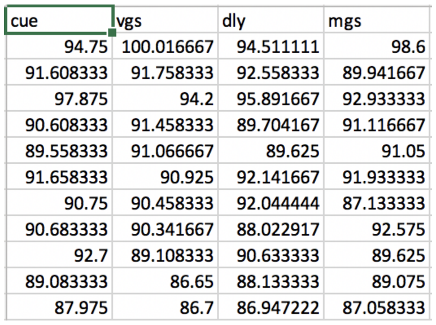


FIGURE 15
Plot that synced based on the lag calculated

To analyze the outcomes further, we pass all the data in two separate csv files representing the x-displacement on pupil and glint. As shown in Figure 16, the dataset above represents pupil and the dataset below represents glint. The difference in the number of rows derived from the fact that glit detection is much faster than pupil detection. But they would be the same in the end.



cue	vgs	dly	mgs
94.741667	104.908333	95.180556	103.483333
89.6	89.691667	92.572222	87.358333
103.733333	94.983333	97.611111	94.933333
89.9	90.333333	88.88125	89.891667
89.458333	91.458333	89.525	91.433333



cue	vgs	dly	mgs
94.75	100.016667	94.511111	98.6
91.608333	91.758333	92.558333	89.941667
97.875	94.2	95.891667	92.933333
90.608333	91.458333	89.704167	91.116667
89.558333	91.066667	89.625	91.05
91.658333	90.925	92.141667	91.933333
90.75	90.458333	92.044444	87.133333
90.683333	90.341667	88.022917	92.575
92.7	89.108333	90.633333	89.625
89.083333	86.65	88.133333	89.075
87.975	86.7	86.947222	87.058333

FIGURE 16
Accurate outcome generated that shows the displacement on five stages.

Even though the tracker can identify five stages from the pupil or left and right directions, due to the asynchronous behaviors and between the original data and the video frames, we could still spot specific errors that existed in the outcomes, which would eventually be a slippery slope without a proper algorithm to adjust it. To further shape the results and improve future analysis accuracy, all the data retrieved for different video files will be collected, manually labeled, and eventually pass into a machine learning algorithm(mostly possibly random forest). Due to the data's repeating patterns, it would be easy for the random forest to detect and eliminate any fault that existed aptly – whether given rise by tracker malfunction, asynchronous behaviors or simply error made by the users.

IMPLEMENTING RANDOM FOREST ALGORITHM IN R

Random Forest and input data

To get the more intuitive results and the best performing modules for precise analysis, Random Forest(A very popular machine learning algorithm) will be implemented to train, evaluate, and generate the data module. As shown in Figure 17, random forest applies bootstrap aggregating(bagging) to tree learners. Given a training set X = x1,....,xn with response Y = y1, …, yn, bagging repeatedly selects a random sample with replacement of the training set and fits trees to these samples. Moreover, to avoid overfitting, the resulting random forest needs tuning to ensure proper functioning. But first, 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 as shown in Figure 18. [7]

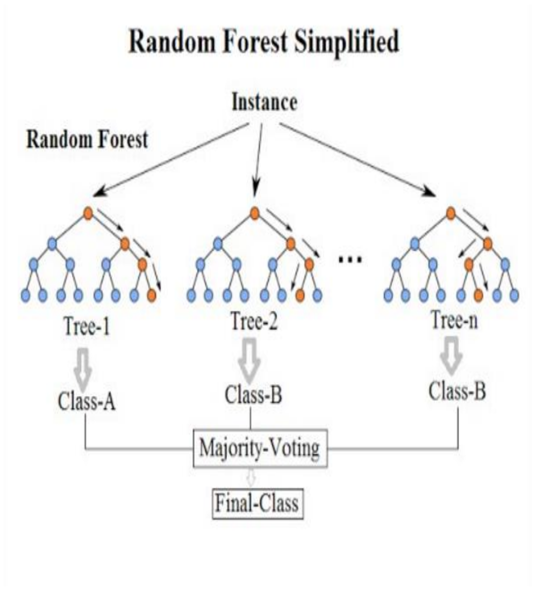


FIGURE 17[7]
Random Forest Machine Learning Algorithm

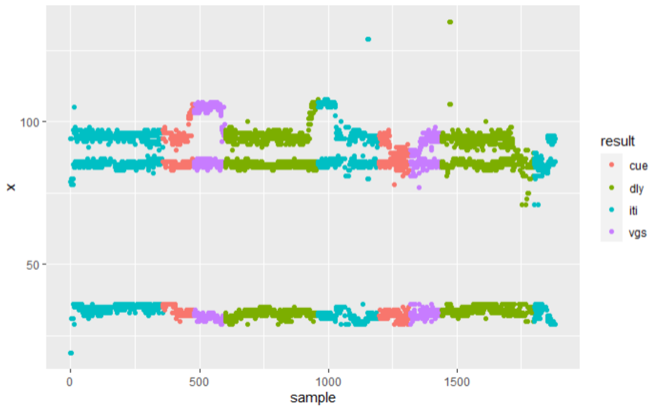
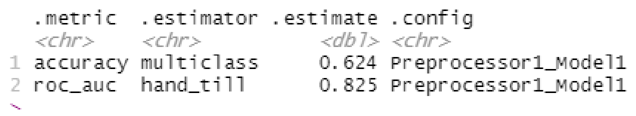


FIGURE 18
Plotting of the input data

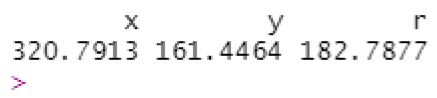
Generating and analyzing machine learning modules

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 data sets for forest tuning. After the successful generation of the Random Forest model, the module is used to evaluate the testing cases, and the results are shown in Figure 18, with ROC value reaching 0.82, which is considered very accurate in terms of performance. To get a better understanding of the whole data, more metrics will be evaluated, including variable importance shown in Figure 19, which shows the weights of each variable, confusion matrix is shown in Figure 20, and the module effectiveness plot is shown in Figure 21.



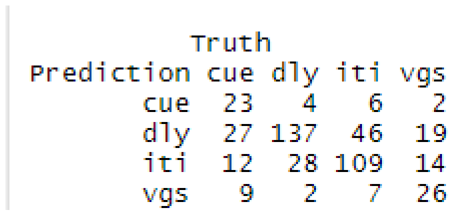
```
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy multiclass     0.624 Preprocessor1_Model1
2 roc_auc  hand_till      0.825 Preprocessor1_Model1
```

FIGURE 18
Showcases of the performance metrics



```
       x        y        r
320.7913 161.4464 182.7877
>
```

FIGURE 19
Variable importances



```
          Truth
Prediction cue dly iti vgs
       cue  23   4   6   2
       dly  27 137  46  19
       iti  12  28 109  14
       vgs   9   2   7  26
```

FIGURE 20
Confusion matrix

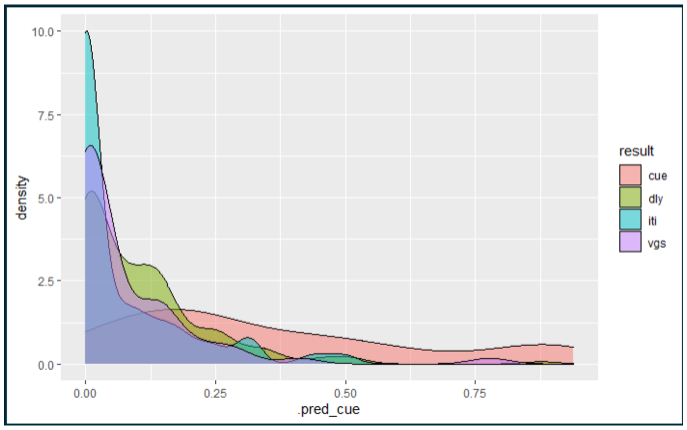


FIGURE 21
Prediction module data visualization

After all the data analysis and forest tuning are completed, a thorough training module would be generated based upon x, y, and r to classify the results that matches onto the stages: cue, vgs, dly, mgs and iti. Right now 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 the model using new data generated every time the app is called. As a result, the tracker would eventually 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 can not handle.

SOURCES

[2] "Lecture 10: Hough Circle Transform", Harvey Rhody, October 11, 2005 [Online]. Available: https://www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf. [Accessed: 3-December-2020]

[3] "Gaussian Blur ", Wikipedia , 31 October 2020[Online]. Available:https://en.wikipedia.org/wiki/Gaussian_blur#:~:text=The%20Gaussian%20blur%20is%20a,each%20pixel%20in%20the%20image. [Accessed: 3-December-2020]

[4] "Canny Edge Detector", Wikipedia , 26 November 2020[Online].Available: https://en.wikipedia.org/wiki/Canny_edge_detector

[5] "Otsu Thresholding", Dr. Andrew Greensted, 17th June 2010[Online], Available: : http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html#:~:text=during%20the%20calculation.-,Otsu%20Thresholding%20Explained,fall%20in%20foreground%20or%20background.

[6]"Tracking with correlation filter", Courseware Wiki, 2018/2/19[online], Available: https://cw.fel.cvut.cz/b172/courses/mpv/labs/4_tracking/4b_tracking_kcf#:~:text=When%20testing%2C%20the%20response%20of,adapts%20to%20moderate%20target%20changes.

[7]"Random Forest ", Wikipedia , 25 October 2020[Online]. Available: https://en.wikipedia.org/wiki/Random_forest

ACKNOWLEDGMENTS

I would like to show my gratitude to my manager in LNCD—Will Foran and professor Finn Calabro for their help in providing valuable insights and necessary help throughout this research.