Robust Eye-Gaze Tracking and Its Application in MRI Imaging

Jiachen Tian, William Foran, and Finnegan J Calabro

***Abstract*—Eye-gaze tracking is widely implemented across numerous areas to study and track observer interaction with scenes and devices. However, noisy visual environments pose a particular challenge to existing eye-gaze tracking when the pupil is barely distinguishable from the background. This research introduces a practical method for tracking eye-gaze under nonideal conditions. We demonstrate the method on data acquired during a high-field MRI scan, in which the MR-compatible hardware resulted in added visual noise. Our method uses customized filtering, preprocessing, and machine learning to track the eye-gaze position despite these limitations while also minimizing processing time and providing immutability to constantly changing noise characteristics in the video.**

***Index Terms*—MRI Imaging, Estimation of eye gaze, machine learning, image processing, gear free.**

I. INTRODUCTION

INFRARED eye-gaze 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 and illuminates the eyes, producing a reflection which can be detected by the camera but will not disturb the users [1]. The reflection allows the pupil's displacement to be tracked and analyzed for the user's precise gaze-position.

Proprietary solutions are provided by most hardware vendors but are not interchangeable and are difficult or impossible to extend. Existing computer vision algorithms should allow gaze tracking to be readily 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.

In high noise environments, real-time tracking of pupil location may be impractical. In such cases, the tracking system may record the infrared camera data as video files and store them for further offline analysis. Here, we describe a novel method for identification of the pupil in high noise environment for rapid, offline video-based analysis.

II. HIGH NOISE EYE GAZE DATASET

In one example of a high visual noise environment, we obtained eye tracking data collected from participants undergoing a high magnetic field (7 Tesla) MR scan (see Figure 3A). Hardware limitations due to the magnetic field rendered the video noisy and low resolution, causing out-of-the box eye tracking software solutions to fail to reliably identify and track the pupil location. In this data set, eye tracking was acquired while subjects performed an oculomotor working memory task, the Memory Guided Saccade (MGS) task. This task consists of five repeated events instructing participants to make expected saccades (shown in Figure 1): (i) Gaze fixation at the center of the screen on cue; (ii) Peripheral saccade to a dot appearing in one of four non-central locations; (iii) Return fixation at the center on cue; (iv) Peripheral saccade to where he or she remembered the image was cued by the disappearance of the central fixation mark; (v) Return fixation to the center of the screen on cue. Accuracy of the saccade at each of these stages can be measured against an expected value to provide a score of each participant's behavior.

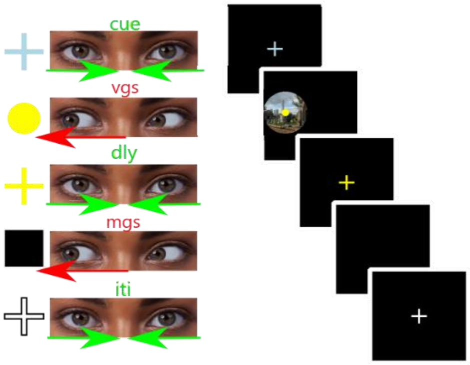

Fig. 1. Demonstration of five instructions for the Memory Guided Saccade task.

III. VIDEO FILE PREPROCESSING IN DETAILS

Our high noise eye tracking approach involves coupling a series of image-based preprocessing steps with Hough transform-based feature identification. Here, we describe the

---

Date of publication May 14, 2020. This work is supported by NIMH grant 5R01 MH067924 to B.L.
Jiachen Tian is with Swanson School of Engineering, University of Pittsburgh, PA 15213 USA (e-mail: tjc990806@gmail.com).
William Foran is with the Department of Psychiatry, University of Pittsburgh, PA 15213 USA (e-mail: foranw@upmc.edu).
Finnegan J Calabro is with the Department of psychiatry, University of Pittsburgh, PA 15213 USA (e-mail: fjc20@pitt.edu).

general approach implemented for these steps. These steps are implemented in python and make extensive use of the OpenCV library [10]. Source code is hosted on GitHub (https://github.com/Tian99/High-Speed-Noise-Tolerant-Eye-Gaze-Tracker).

A. Image Preprocessing

We start with a gray scale image (pixel values 0 to 255). As shown in Figure 2, noise is prevalent throughout the entire frame. Furthermore, this noise changes across different image frames over time, making it hard for the algorithm to determine the location of the pupil and glint. To reduce noise and to generate more reliable frames for testing, we first implemented a Gaussian blur to smooth the frames extracted from video file and to remove any unnecessary noise in the frame. Gaussian blur can be represented by (1) [3].

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

A convolution matrix is built with the formula, which applies to the original image. Importantly, Gaussian filters have the possibility to preserve features necessary to detect glint, which has a small size. Even though the blurring factor can be kept identical across all the frames in one video file, a constant value cannot be robust to noise that varies across frames. Our strategy for optimizing blurring parameters is described later.

Next, we threshold the image to obtain a binarized representation of the frame. Thresholding after blurring can vastly reduce the chance of detecting unnecessary edges. Like blurring, two parameters are required to define a range (lower threshold, higher threshold), which varies depending on image size, illumination, and other sources of noise for every video file. Threshold parameters are explored identically to blurring parameters, also described later.

Canny edge detection is a technique to extract useful structural information from different visual objects. The edge consists of two directions: x and y. Parameterization of the canny filter is more robust than blurring factors or threshold factors: predefined canny parameters are sufficient when the image is optimally thresholded. Therefore, to optimize the runtime efficiency, canny factors would be a constant value define by users.

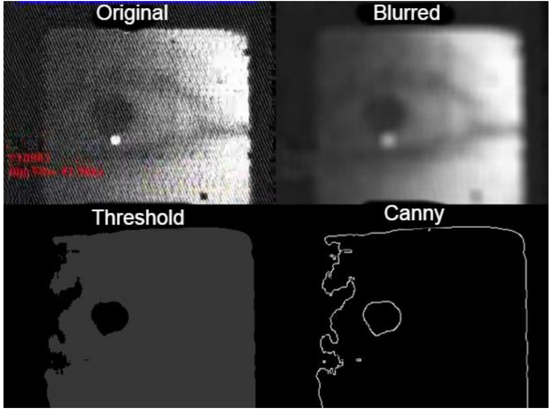

Fig. 2. Three stages filtering with constant predefined parameters showing three stages of image preprocessing for one of the frames from videos taken by the MRI camera. Including original frame, blurred frame, threshold frame and canny frame.

B. Pupil Identification by Hough Transform

The Hough Transform [4] is a powerful computer vision tool to detect lines, circles, ovals, or other polygons in an image. We focus on circle detections as part of the research goal to identify the pupil. The Hough Circle Transform uses three coordinates that define a circle in a 2D space, which can be described as (2) [2].

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

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 (3) and (4) [3].

$$x = a + r\cos(\theta) \tag{3}$$
$$y = b + r\sin(\theta) \tag{4}$$

as shown in Fig 3. To decide which location forms a circle, we iterate through all the pixel positions in a rectangle specified by the user, represented as (x, y). For every edge pixel detected, we iterate through potential polar angles from 0 to 360 for the potential radius defined based on the average radius of the pupil. 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 stored. However, due to its voting mechanism, the Hough Transform algorithm could still be affected by noises which are prevalent in our video datasets. This is mitigated by the aforementioned preprocessing steps (blurring, image thresholding, and canny image transformation).

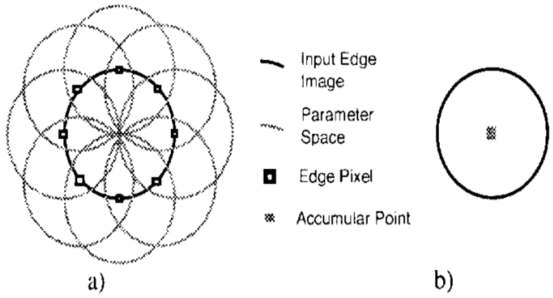

Fig. 3. Hough Transform Detection where on the left is the demonstration of iterations under the polar coordinates whereas on the right is the detected circle with the center having the most votes. [3]

C. Tracking Model Parameterization

Given variations in image size, illumination, and noise patterns, the optimal set of preprocessing steps and parameterization above may vary both within and across datasets. Thus, we use a method to identify optimal preprocessing parameters, in particular for blurring and thresholding.

The blurring factors are determined by iterating through a “pre-test” subset (the first 5000 frames), varying blurring factors within a range defined by users. The optimal parameters should be frequently repeated across frames. When there are differences between frames, it should be small since pupil displacement between frames is minimal. Initial blurring factors are determined by finding the pre-tested iteration with the smallest standard deviation, as shown in Figure 5.

Next, to retrieve the thresholding factors, the tracker is implemented across all frames to be tracked, iterating across a range of different threshold factors from 0 to 255 while keeping the blurring and canny factors constant. When other factors are held constant, the threshold parameters that results in the largest numbers of circles would be considered the best, computed based on the voting mechanism of the Hough Transform (see Figure 4).

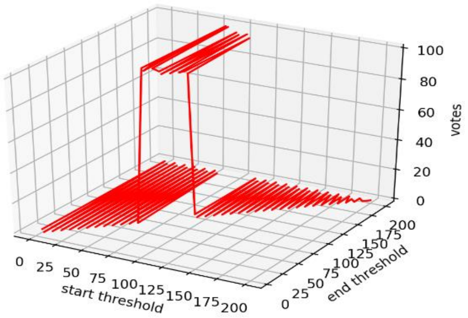

Fig. 4. Plot showing votes under ideal blurring factors and different thresholding factors where blurring factors represent a certain range from a total range of 0 to 255 as shown by X and Z coordinate in the graph. Y coordinate represents number of successful votes for each iteration.

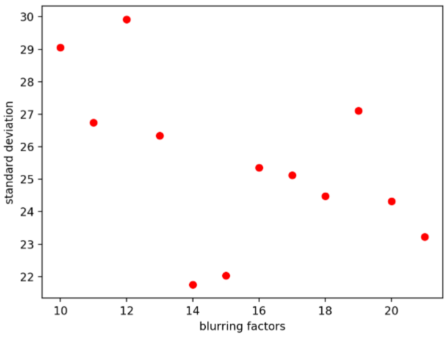

Fig. 5. Plot showing standard deviation under changing threshold factors for different blurring factors where the lowest standard deviation represents the best blurring factors.

D. Preprocessing for Glint

Our primary approach is to track based on identification of the pupil itself. However, point of gaze estimates could be improved by identifying the infrared glint that appears from reflection of the IR illuminator [9]. The preprocessing steps for glint detection are nearly identical to pupil preprocessing. However, blurring may be unnecessary due to the size of the glint. Thus, for glint we utilize an automated threshold tracker alone. Compared to the pupil, glint is more distinguishable from the background and easier for the corresponding algorithms to recognize. We implemented an Otsu’s Thresholding method to 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 is 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.

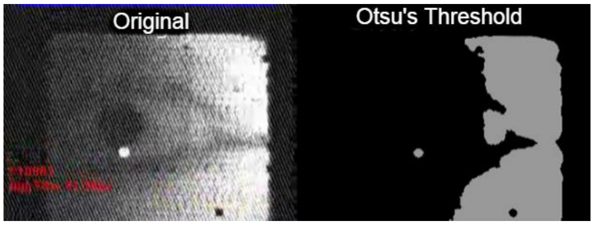

Fig. 6. Finding the glint using Otsu’s Threshold where it automatically generates the best threshold range represented by (lower threshold, higher threshold) that represents the best demonstration of glint.

E. 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 fails to uniquely identify the pupil and/or glint. 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 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 tracker to adapt to moderate target changes [6].

Despite its incapability of tracking the precise location, the KCF tracker can, in turn, be used to filter out the wrong values (outliers) from the Hough Transform. Rectangular boxes selected by users are 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, drawing over a derived image representing a fully open eye, and the results would be dynamically drawn on the right. We generate the open eye image by iterating and summing over pixel (necessarily dark/low) values in an image.

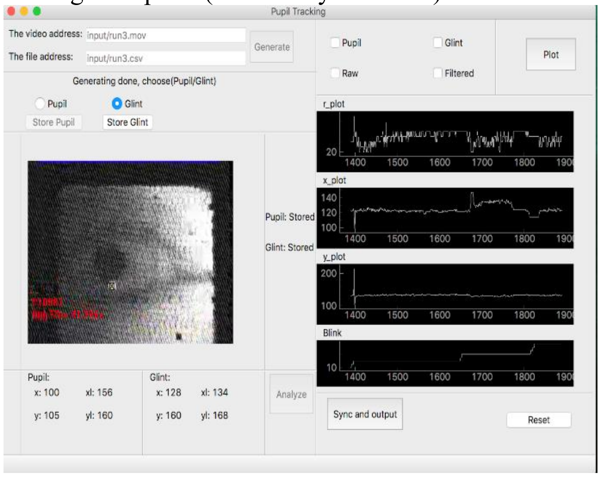
The video address: input/run3.mov
The file address: input/run3.csv
Generating done, choose(Pupil/Glint)
○ Pupil ● Glint
Pupil: Stored
Glint: Stored
Pupil: x: 100 xl: 156 y: 105 yl: 160
Glint: x: 128 xl: 134 y: 160 yl: 168
☐ Pupil ☐ Glint
☐ Raw ☐ Filtered

Fig. 7. User interface that enables selections and animation of tracking results. Users crop out pupil and glint areas on the right of the interface and the plot (radius, x-coordinate, y-coordinate, blinking rate) would show dynamically in real-time on the right of the interface.

F. *User-Supplied Intervention*

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's 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 half the original glint size. All the glint movements in the video would be included. Therefore, the drawback 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 more filters, accurate results could still be retrieved.

IV. TRACKING RESULTS

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. For simplicity, 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. Initial sampling rate is 60Hz. Time course plots have been smoothed for visualization.

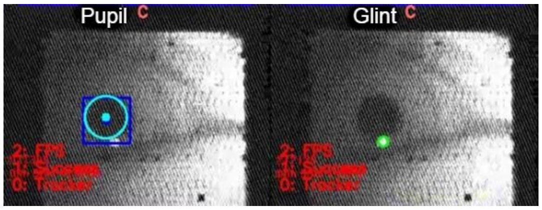

Fig. 8. Tracking results plotted onto original frame for both pupil and glint. On the pupil, circle represents result of Hough Transform and rectangle represents result of KCF tracker. If one tracker failed, the other one would take over. On the glint, it shows the case when KCF tracker failed and the Hough Transform takes over.

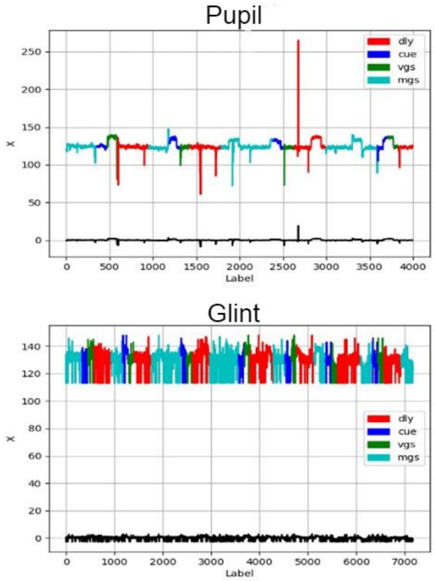

Fig. 9. Unfiltered position plot showing the motion in x direction for both pupil and glint. The coloring is used to match on the input data with the tracking output data. Black line underneath shows z-scores which is crucial for the later data filtering.

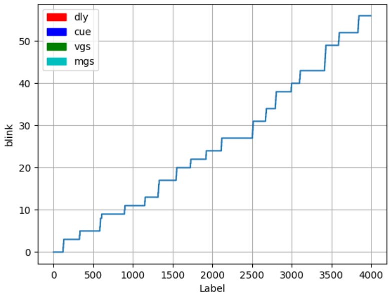

Fig. 10. Target Blinking rate plot that is not labelled by colors. X axis shows the frame number representing when the blink happens and Y axis shows the count of blinks up to a certain number of frames.

A. *Result Data Filtering*

The first step of filtering the resulting data is to compare the radius and position of glint and pupil so that the data makes sense when putting them together, i.e., the radius of the glint should never be bigger than that of the pupil. After completing this first pass, we implement 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). 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.

$$z = \frac{datapoint - mean}{standarddeviation} \qquad (5)$$

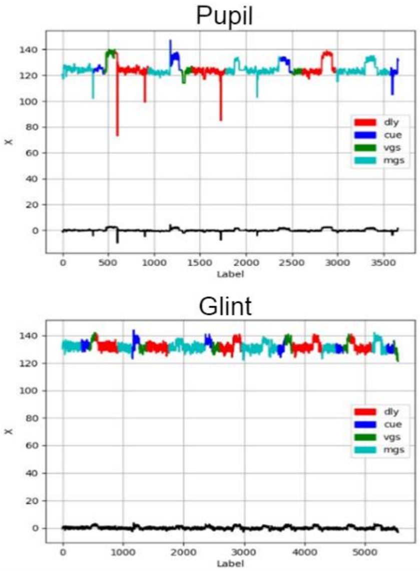

Fig. 11. Filtered position plot showing the motion in x direction. Black line underneath shows Z-scores for the data. Comparing to Figure 9, the vast reduction of outliers in both pupil plot and glint plot shows the effectiveness of data filtering.

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 figure 12 [8]. 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.

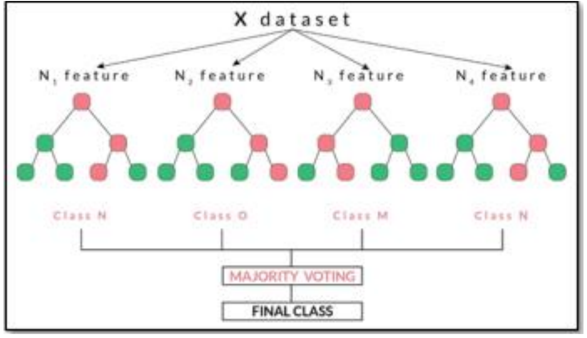

Fig. 12. Random Forest structure where datasets are broken down into a different decision trees each performing separate classifications on the input data. [8]

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 [11] 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 Table I, 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 Table II, which shows the weights of each variable.

TABLE I
RANDOM FOREST PREDICTION ACCURACY

.metric	.estimator	.estimate	.config
Accuracy	Multiclass	0.624	Preprocessor_model
Roc_auc	Hand_till	0.825	Preprocessor_model

TABLE II
VARIABLE IMPOERANCE

x	y	r
320.7913	161.4464	182.7877

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 labeled. 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 low quality videos acquired in a noisy environment. 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.

ACKNOWLEDGMENT

I would like to show my gratitude to professor Dallal, Hassan Sayed Ahmed for his help in providing valuable insights throughout this research. This work is supported by NIMH grant 5R01 MH067924 to B.L.

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] Henriques J. F., Caseirio R., Martins P., Batista J. High-Speed Tracking with Kernelized Correlation Filters. IEEE Trans on PAMI 37(3):583-596. 2015

[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] 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.

[9] M. Gneo, “A free geometry model-independent neural eye-gaze tracking system,” p. 15, 2012.

[10] Bradski G. “The OpenCV Library.” Dr Dobb’s Journal of Software Tools. 2000;

[11] Kuhn M, Wickham H. “*Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*” https://www.tidymodels.org. 2020