EyePy: A Computationally Efficient and Open-Source Solution for Webcam-Based Eye Tracking

Chenkai Zhang

September 24, 2024

Abstract

Eye tracking technologies are technology that predicts a user's gaze location, and its application ranges from marketing to neuroscience. However, current solutions often rely on expensive hardware, limiting the access to such technologies from researchers and students. This paper introduces "EyePy," a computationally efficient, webcam-based eye tracking solution utilizing machine learning and computer vision techniques to offer a cost-effective alternative. EyePy uses traditional machine learning methods to predict gaze locations effectively in real time. It differs from previous webcam-based alternatives in that it does not use CNN, which is computationally expensive, and accounts for head orientation, enhancing the accuracy of gaze estimation. EyePy achieves competitive performance compared to previous more computationally heavy systems without sacrificing precision of the model. EyePy aims to democratize eye tracking technology by offering an open source and easy to deploy solution.

Keywords: Machine Learning, Eye Tracking, Webcam-Based Eye Tracking

Contents

1 Introduction

Eye Tracking (ET) is a technology that predicts a user's gaze position using a combination of hardware and software. ET has found diverse applications ranging from marketing to neuroscience. In marketing, ET is used to help marketers have a better understanding of consumer behavior by tracking their focus. However, ET is especially prominent in neuroscience and cognitive research (Popa et al., 2015). It's shown good performance in assessing cognitive abilities, revealing distinct types of mnemonic information (Hannula et al., 2010) There has been research on using ET to diagnose traumatic brain injuries, concussions (Samadani et al., 2015) (Snegireva, Derman, Patricios, & Welman, 2018), dementia (Hutton, Nagel, & Loewenson, 1984) (Mengoudi et al., 2020), and autism (Falck-Ytter, Bölte, & Gredebäck, 2013) (Guillon, Hadjikhani, Baduel, & Rog é, 2014).

However, professional eye trackers are a significant barrier to entry for aspiring researchers. Their cost and complicated setup could deter researchers from using ET methods. With the demand for a simpler solution, there are various products attempting to use web cameras (webcams) for gaze tracking. Sadly, current implementations of webcam-based gaze tracking often come with its own set of limitations, including inferior accuracy and being computationally expensive.

This study presents a webcam-based gaze tracking solution that achieves good accuracy while remaining computationally efficient. It achieves this by also taking in head rotation into account for prediction. The vision for this project is to have a simple to deploy solution that will help push ET research and its adoption in other fields forward.

2 Background and Related Works

2.1 Traditional Eye Tracking Methods

There are two main traditional methods to eye tracking. Optical tracking and Electrooculography.

2.1.1 Optical Tracking

Optical tracking methods work by first shining infrared light at the eye then processing the corneal reflection and pupil's position relative to the eye socket. This is one of the most robust and wide spread eye tracking methods. However, such systems often require complicated hardware setups and sometimes also require head stabilization device, such as head mounts or chin rests.

2.1.2 Electrooculography (EOG)

Electrooculography (EOG) is another traditional eye tracking method. It works by attaching electrodes around the eye and utilizing the fact that the retina is electrically polarized to calculate the eye orientation. It is responsive but has less precise gaze predictions. Attaching electrodes around the eye is also invasive to a extent.

2.1.3 Limitations of Traditional Methods

Although those traditional methods, especially optical tracking, have proven its effectiveness in past real world studies, the requirement for eye tracking hardware are significant barriers to entry for unpaid researchers and students looking to research in ET areas (Hessels & Hooge, 2019). Costing from $200 to over $10,000, the price represents a considerable expenditure for researchers.

2.2 Webcam-Based Eye Tracking Methods and Its Limitations

One of the most practical implementation of webcam-based eye tracking is Search Gazer (Papoutsaki, Laskey, & Huang, 2017). It is also a big inspiration for this project. However, it's written in JavaScript, which is less suited for data analysis and is mainly geared towards website usage. Additionally, requiring a webpage to conduct studies makes the setup overly complicated for simple research projects.

There are various studies that uses the CNN architecture for gaze prediction. One study by Krafka achieved good accuracy with such prediction models (Krafka et al., 2016), Another study by Gudi shows the efficiency achieved with a webcam-based eye tracking system using a similar approach. (Gudi, Li, & van Gemert, 2020) These studies achieves reasonable performance, however with the computationally heavy nature of deep learning models, they could face challenges in real world usage.

While there are online platforms for ET based on webcams, those implementations often come with cumbersome setup processes and are sometimes platform dependent (*GazeRecorder*, 2024). What's more some implementations require a subscription to use (*Real Eye*, 2024) (*Eyeware Beam*, 2024), going against the goal of reducing friction for ET research in the first place, deterring researchers who are looking for a simple and straightforward process.

3 Method

3.1 Environment Setup

The development and testing environment includes the following packages:

- Python 3.12.4
- OpenCV 4.10.0
- NumPy 1.26.4

- Pygame 2.6.0
- Joblib 1.4.2
- scikit-learn 1.5.1
- Matplotlib 3.9.1
- scikit-image 0.24.0
- dlib 19.24.4

3.2 Implementation

The implementation of the gaze tracking system in this study includes three stages: facial data processing, training the eye tracking model, and gaze prediction.

3.2.1 Facial Data Processing

This process begins with capturing the image of the face and feeding it through dlib's facial landmark detector by Adrian Rosebrock (Rosebrock, 2017).

After that, we use uses the GazeTracking library by Antoine Lamé (Lamé, 2022) to extract the relative pupil position within the eye. The library has been modified to directly take in the frame and facial landmarks instead of calculating facial landmarks again to reduce redundant calculation.

Then the rotation vector of the head is calculated using the Efficient Perspective-n-Point (EPnP) algorithm within OpenCV. Perspective-n-Point (PnP) is the problem of estimating the pose of an object given a set of 3D points and their corresponding 2D projections. This method projects the 3D points onto a 2D canvas to calculate the pose of the 3D object. The core of this algorithm can be formulated as an optimization problem:

$$\min_{R,t} \sum_{i=1}^{n} \|\text{proj}(K(R\mathbf{X}_i + t)) - \mathbf{x}_i\|^2$$

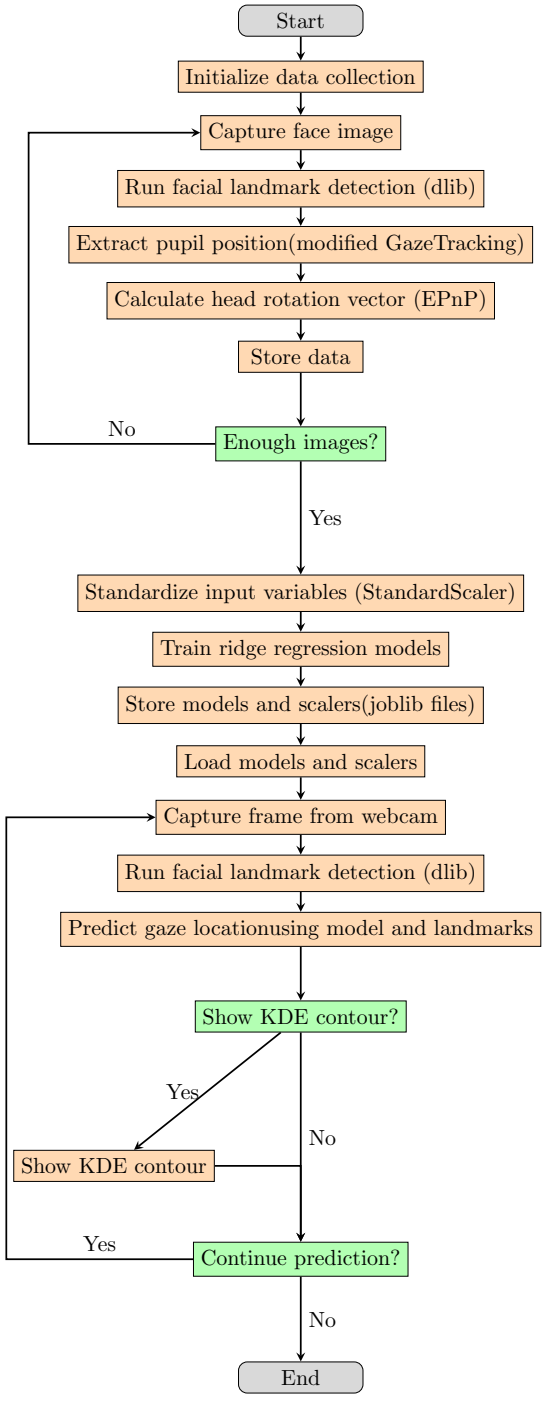
The predefined 3D coordinates in this application include locations of the nose tip, chin, both outer eye corners, and the outer mouse corners. The selected facial landmarks and 3D coordinates are fed into the function, and the function outputs the pitch (tilt up or down) and yaw (turn left or right) of the head.

3.2.2 Model Training

The model used in this study is ridge regression. The formula of ridge regression is given by:

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$$

Ridge regression functions like ordinary least squares regression but includes a penalty for large coefficient through the regularization parameter y. This makes it less likely to overfit and produce more stable results. In this study, ridge regression is used because the noise in the captured data would likely cause ordinary linear regression to overfit.



r1

Figure 1: Implementation Flowchart

Two separate ridge regression models are trained. One predicts the x coordinate using the head rotation vector and relative pupil position in the x-axis. The other predicts the y coordinate using the rotation vector and relative pupil position in the y-axis. The alpha could be changed and is defaulted to 1.

Greater weight is given to pupil position than head rotation. This weight is defaulted to 3:1 and could be adjusted accordingly.

The input variables are standardized using StandardScaler from scikit-learn before they are fed into the model, This is done by subtracting the mean and dividing by the standard deviation. The variables are standardized because the rotation vector and horizontal and vertical ratios have different magnitudes of scale. Doing this ensures that both features contribute equally to the model.

The trained model and scaler are then stored as joblib files.

3.2.3 Gaze Prediction

The gaze prediction begins with loading the joblib files of the model and scaler. The program captures the frame from the webcam, pass it through dlib's facial landmark detector, and use the model and the landmarks to predict the gaze location in real time.

The program includes two optional features, one is to show the kernel density estimation contour at 60% confidence level for the predicted gaze points over the last 0.5 second. The other is to pass the predicted gaze points through a Kalman filter. Both functions help to better visualize the user's gaze behavior.

3.3 Validation Technique

Two validation methods are used to test the performance of the gaze tracking model, including stability and precision. The model used for validation is trained with 10 calibration points: two points at each corner of the screen and two in the center

3.3.1 Stability Test

After calibration, the subject is instructed to focus on a circle in the center of the screen for 10 seconds. During this period, the gaze location is predicted and recorded. After that,

the Standard Deviation (STD), Mean Absolute Deviation (MAD), Bivariate Contour Ellipse Area (BCEA), and Signal-to-Noise Ratio (SNR) of the gaze points are calculated.

3.3.2 Precision Test

For the precision test, the screen is split into a 5x5 grid. A random grid light up every 4 second. After a 2 second delay, the subject's gaze data is recorded for 1 second. The program checks whether the mean position of the points predicted during that second is within the rectangle, and metrics similar to those in the static test are also calculated. That process is repeated for 20 times, with a 1 second pause between each iteration.

3.4 Degree Calculation

To translate from the errors in pixel to errors in field of view in degrees for better comparison with other studies, this formula is used to calculate the field of view angle for one pixel:

$$\text{Angle (degrees)} = 2 \times \arctan\left(\frac{\text{Pixel size}/2}{\text{Viewing distance}}\right)$$

4 Results

The performance of the gaze tracking model is assessed with the methods detailed in Section 3.3

4.1 Results of Stability Test

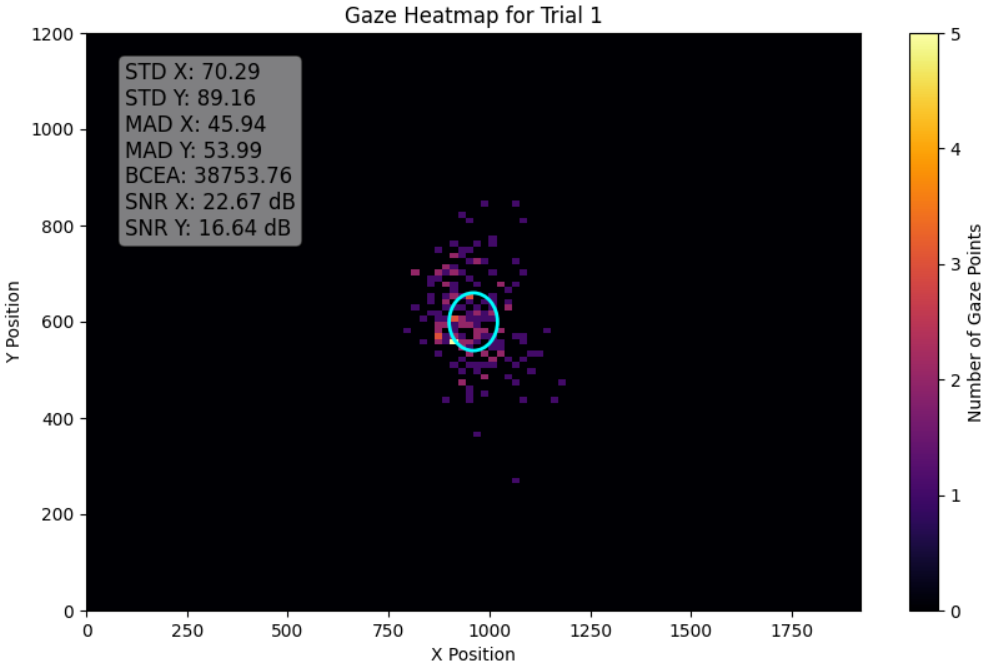


Figure 2: Heatmap

As shown in the Figure 2, the model achieved good accuracy, with Standard Deviation (STD) in the x direction being 70.29 pixels and STD in the y direction being 89.16 pixel. The test being conducted on a 16 inch screen with a 1920x1200 pixel resolution, which means that the STD for x and y respectively is about 1.26 and 1.60 centimeters. The subject sat about 50 cm away from the screen during the process, using the formula from Section 3.4, We can obtain the standard deviation of gaze points in degrees of visual angle to be 1.45° for x and 1.83° for y.

4.2 Result of Precision Test

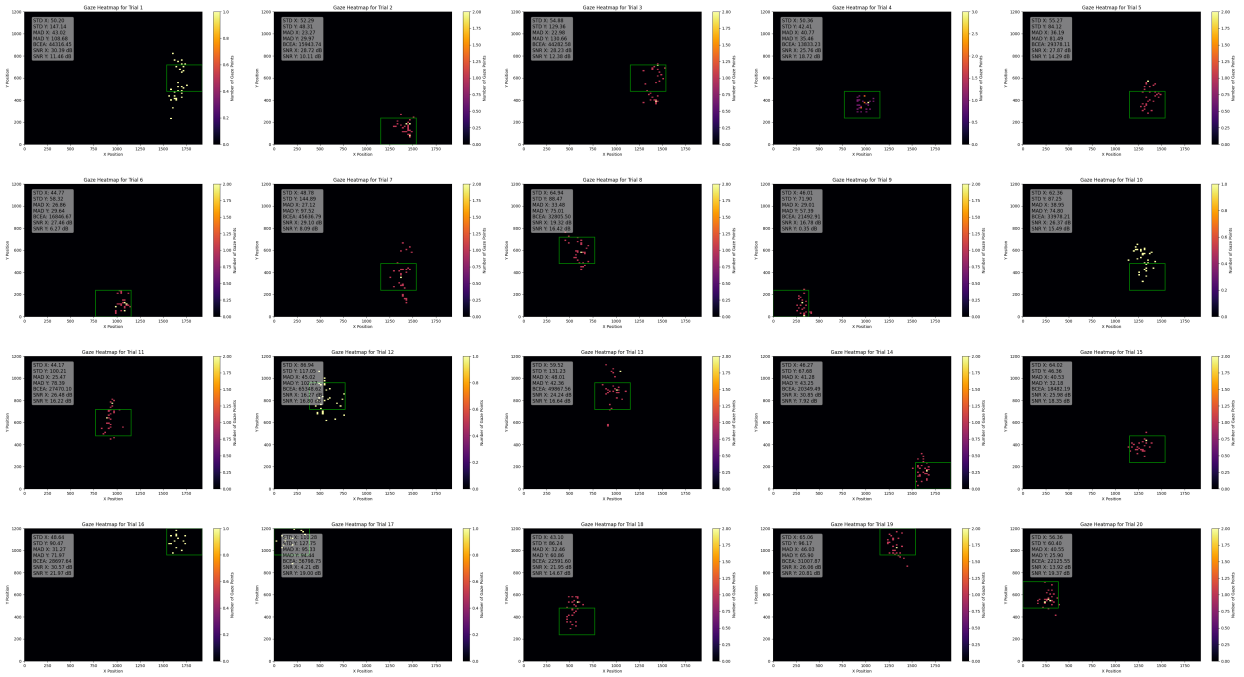


Figure 3: Heatmap

As shown in the Figure 3 The mean of the predicted points ended up in the rectangle 19 out of the 20 trials. Demonstrating good precision, the system achieved a 95% accuracy rate, surpassing the 78% rate reported in previous studies (Zheng & Usagawa, 2018). The groupings also demonstrate similar stability displayed in the previous section, proving the robustness of the model.

4.3 Processing Speed

The gaze prediction model runs at 30 frames per second (fps) on 13th Gen Intel i9-13900H running on a single core. This result surpasses previous implementations using CNN which achieved 10-15 fps (Krafka et al., 2016) (Gudi et al., 2020). Additionally, this test was conducted without optimization for Python's Global Interpreter Lock (GIL) and without utilizing GPU acceleration. Future tests will explore multi-core and GPU-based optimiza-

tions to enhance performance further.

5 Conclusions and Future Work

This study showcase a webcam-based eye tracking system with both speed and accuracy surpassing previous implementations. The study achieves good computation efficiency through the use of a simple ridge regression model for the pupil's location within the eye and the head rotation vector, which no previous study has taken into account.

This study shows the validness of traditional algorithms in the field of webcam-based eye tracking, with potential for commercial application thanks to the gained performance.

However, there's still much left to be done. As stated above, the model does not take advantage of parallel processing or GPU acceleration, which significantly limits its performance. Also, a future direction for the project would be to replace dlib with MediaPipe, a more modern and flexible solution for facial landmark detection that performs better especially for real time applications.

The source code of the model is hosted on GitHub `https://github.com/ck-zhang/EyePy`.

References

Eyeware beam. (2024). https://beam.eyeware.tech/. (Accessed: 2024-09-18)

Falck-Ytter, T., Bölte, S., & Gredebäck, G. (2013). Eye tracking in early autism research. *Journal of neurodevelopmental disorders*, *5*, 1–13.

Gazerecorder. (2024). https://gazerecorder.com/gazerecorder/. (Accessed: 2024-09-18)

Gudi, A., Li, X., & van Gemert, J. (2020). Efficiency in real-time webcam gaze tracking. In *Computer vision–eccv 2020 workshops: Glasgow, uk, august 23–28, 2020, proceedings, part i 16* (pp. 529–543).

Guillon, Q., Hadjikhani, N., Baduel, S., & Rog é, B. (2014). Visual social attention in autism spectrum disorder: Insights from eye tracking studies. *Neuroscience & Biobehavioral Reviews*, *42*, 279–297.

Hannula, D. E., Althoff, R. R., Warren, D. E., Riggs, L., Cohen, N. J., & Ryan, J. D. (2010). Worth a glance: using eye movements to investigate the cognitive neuroscience of memory. *Frontiers in human neuroscience*, *4*, 166.

Hessels, R. S., & Hooge, I. T. (2019). Eye tracking in developmental cognitive neuroscience–the good, the bad and the ugly. *Developmental cognitive neuroscience*, *40*, 100710.

Hutton, J. T., Nagel, J., & Loewenson, R. B. (1984). Eye tracking dysfunction in alzheimer-type dementia. *Neurology*, *34*(1), 99–99.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2176–2184).

Lamé, A. (2022). *Gazetracking.* https://github.com/antoinelame/GazeTracking. GitHub.

Mengoudi, K., Ravi, D., Yong, K. X., Primativo, S., Pavisic, I. M., Brotherhood, E., ... Alexander, D. C. (2020). Augmenting dementia cognitive assessment with instruction-less eye-tracking tests. *IEEE journal of biomedical and health informatics*, *24*(11),

3066–3075.

Papoutsaki, A., Laskey, J., & Huang, J. (2017). Searchgazer: Webcam eye tracking for remote studies of web search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval* (pp. 17–26).

Popa, L., Selejan, O., Scott, A., Mureşanu, D. F., Balea, M., & Rafila, A. (2015). Reading beyond the glance: eye tracking in neurosciences. *Neurological Sciences*, *36*(5), 683–688.

Real eye. (2024). `https://www.realeye.io`. (Accessed: 2024-09-01)

Rosebrock, A. (2017). *Facial landmarks with dlib, opencv, and python.* `https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`. (Accessed: 2024-09-18)

Samadani, U., Ritlop, R., Reyes, M., Nehrbass, E., Li, M., Lamm, E., ... others (2015). Eye tracking detects disconjugate eye movements associated with structural traumatic brain injury and concussion. *Journal of neurotrauma*, *32*(8), 548–556.

Snegireva, N., Derman, W., Patricios, J., & Welman, K. (2018). Eye tracking technology in sports-related concussion: a systematic review and meta-analysis. *Physiological measurement*, *39*(12), 12TR01.

Zheng, C., & Usagawa, T. (2018). A rapid webcam-based eye tracking method for human computer interaction. In *2018 international conference on control, automation and information sciences (iccais)* (pp. 133–136).