Research Report

Application of event camera on near-eye gaze tracking

Zeyu Li

Introduction

This project is mainly about applying event camera on eye tracking, and my work is based on the Angelopoulos [1] 's work *Event Based, Near Eye Gaze Tracking Beyond 10,000 Hz*, and did some improvements.

Background

Eye tracking is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head. It has many applications, such as human-computer interaction in VR/AR, marketing, research of medicine and psychology. Nowadays cameras are usually used as eye tracker devices. There are many requirements for the devices, for example, it needs a high frequency and low latency, and for a portable near-eye tracker, such as in VR/AR, it needs to be small size, low power consuming and low bandwidth. So these cameras are usually need a trade-off between resolution, frame-rate, and power. Most expensive high-end eye tracking systems resolve this using high-speed cameras, producing huge data with customized protocols and interfaces to maximize bandwidth. Meanwhile, they are large and power hungry. And in fact, most of this data is redundant. Only the pupil moves, while most of the image does not change. So we considered to use a new type of camera called event camera to perform better portable near-eye tracking.

In short, event camera catches the change of light in each pixel. An event camera output an event once a pixel detecting the bright changes more than a preset threshold, and if there is no change, it will keep silent.

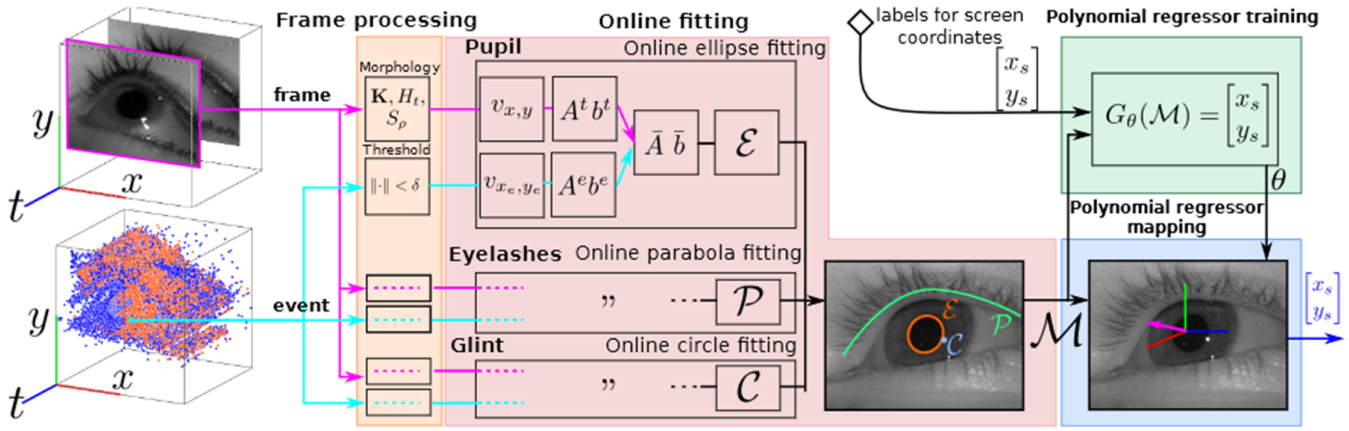
Compared with traditional camera, the event camera has many advantages:

1. A high frequency and low latency at microsecond resolution, and low motion blur. Since it doesn't need to accumulate and process frame, it outputs event stream, which can react very quickly.
2. High dynamic range. Since it only catches the change of light, instead of absolute light. So it can still performance well in a dark or over bright sense.
3. Low power consumption and low data bandwidth requirement. event data is efficient and sparse, since only scene changes are recorded.

But it's also not that perfect. For example, since it's very sensitive, there is lots of noise, which may be hard to process.

Features detecting and tracking is a popular application of the event camera, and eye tracking is one important part of that direction. We can see that, compared with trad camera, an event camera is a good fit for eye tracking sensors in AR/VR headsets, since they fulfill key requirements on power and latency.

My work is based on this paper: Angelopoulos [1] 's work *Event Based, Near Eye Gaze Tracking Beyond 10,000 Hz*.



In summary, it combined events and standard frames from a hybrid event camera to perform eye tracking. The frames were used to initialize models of parametric ellipse pupil, parabola eyelid, and circle corneal glint respectively. Events found within a preset distance of the parametric model fitted by frame were then used to asynchronously update the models between frames. Finally, it trained a regressor to map pupil centers to gaze points on the screen. It also contributed a database containing gray frame, event stream, and target screen pixels. The result is good, it reached 10,000 Hz update rate, good accuracy of Pupil Tracking and gaze mapping. It also provided a dataset contains near-eye gray frames and events.

Methods in Paper

Ellipse Fit for Pupil

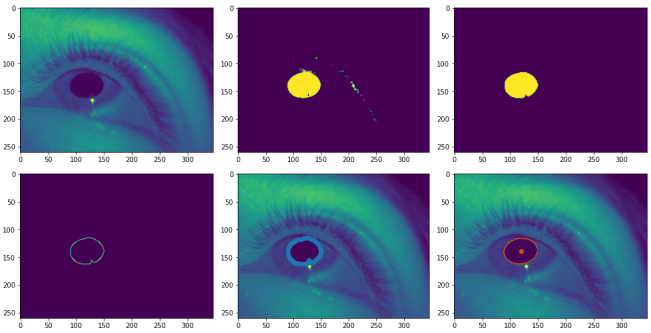
$$E_{\mathscr{E}}(x,y) = 0$$
$$\text{with} \quad E_{\mathscr{E}}(x,y) = ax^2 + hxy + by^2 + gx + fy + d$$

Frame Process

$$\mathscr{D}_{\text{img.}} = \left\{(x,y) \mid \mathbf{K}\left(\mathrm{H}_{\boldsymbol{\theta}}(I) \circ S_{\sigma}\right)(x,y) = 1\right\},$$

where H_θ is the unit step function shifted by θ used for thresholding; "$\circ$" denotes morphological opening; S_σ is its structuring element, a discretized circle parameterized by its radius σ ; and K is a binary edge detection function.

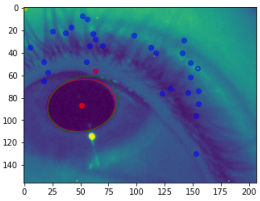
0. Original frame
1. Cut some percentages edge of frame
2. Apply threshold function
3. Apply morphological opening with circles
4. Apply Canny edge detector
5. Calculate and update ellipse parameters



Event Process

$$\mathscr{D}_{\text{evt.}} = \left\{(x,y) \mid \|P_{\mathscr{E}}\big((x,y)\big) - (x,y)\|_2 < \delta\right\},$$

Select event points by distance to the ellipse.



Ellipse Fit

$$\mathscr{E}^* = \underset{\mathscr{E}\in\mathbb{R}^5}{\arg\min} \sum_{(x,y)\in\mathscr{D}_{\text{img.}}} E_{\mathscr{E}}(x,y)^2$$

$\bar{A} = \text{Id}_{5\times5}, \bar{b} = \vec{0}_5$ ▷ Init. $\bar{A}$ and $\bar{b}$
$\bar{A}_{\text{inv.}} = \bar{A}^{-1} = \text{Id}_{5\times5}$ ▷ If using SMW init. $\bar{A}_{\text{inv.}}$

for all $(x,y) \in \mathscr{D} = \mathscr{D}_{\text{evt.}} \cup \mathscr{D}_{\text{img.}}$ **do**
  $v_{x,y}^{\mathsf{T}} \leftarrow (x^2, xy, y^2, x, y)$ ▷ Eq. (13)
  $A \leftarrow \sum_{(x,y)\in\mathscr{D}} v_{x,y}\, v_{x,y}^{\mathsf{T}}$ ▷ Eq. (11)
  $b \leftarrow \sum_{(x,y)\in\mathscr{D}} v_{x,y}$
if $|\mathscr{D}| > 1$ **then** ▷ Batch update (frame/multiple evts.)
  $\bar{A} \leftarrow \gamma\bar{A} + (1-\gamma)A$ ▷ Eq. (7)
  $\bar{b} \leftarrow \gamma\bar{b} + (1-\gamma)b$
  $\bar{A}_{\text{inv.}} \leftarrow \bar{A}^{-1}$ ▷ Full-rank inversion
  $\mathscr{E} \leftarrow \bar{A}_{\text{inv.}}\,\bar{b}$

Parabola Fit for Eyelid

$$\begin{aligned} & E_{\mathscr{P}}(x,y) = 0 \\ \text{with} \quad & E_{\mathscr{P}}(x,y) = a'y^2 + g'y + d' - x \end{aligned}$$

Frame Process

$$\begin{aligned} \mathscr{D}'_{\text{img.}} = \Big\{(x,y) \mid\ & (x,y) \in \text{HarrisCorner} \circ \text{clip}(I, t_1, t_2), \\ & \text{and } \|(x,y) - (x_e, y_e)\|^2 < \rho', \\ & \text{and } y < \frac{\text{rows}}{2} \Big\} \end{aligned}$$

Process clip [t1, t2] and Harris corner detector, select by distance to pupil center, and select only upper half points.

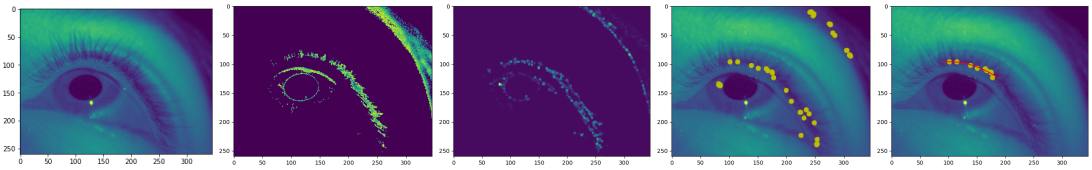
0. Original frames

1. Clip the pixels between some value

2. Apply Harris corner detector, select some percentages high value points

3. Select points in the upper half side of image

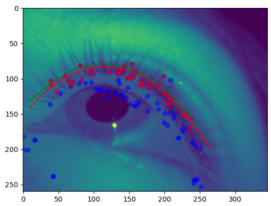
4. Calculate and update parabola parameters. The open side of parabola was limited as positive.



Event Process

$$\mathscr{D}'_{\text{evt.}} = \left\{ (x,y) \mid |E_{\mathscr{P}}((x,y))| < \delta \right\}$$

Select event points by distance to parabola curve (parabola function value).



Parabola Fit

$$\mathscr{P}^* = \underset{\mathscr{P}\in\mathbb{R}^3}{\arg\min} \sum_{(x,y)\in\mathscr{D}'} E_{\mathscr{P}}(x,y)^2$$

solution is simply $\mathscr{P}^* = A'^{-1} b'$ with

$$A' = \sum_{(x,y)\in\mathscr{D}_{\text{img.}}} v'^1_{x,y} v'^{1\top}_{x,y}, \quad b' = \sum_{(x,y)\in\mathscr{D}_{\text{img.}}} v'^2_{x,y} \tag{11}$$

in which $v'^1_{x,y}$ is now:

$$v'^1_{x,y} = (y^2, y, 1) \tag{12}$$

and $v'^2_{x,y}$ is different (due to the asymmetry in x and y):

$$v'^2_{x,y} = x\, v'^{1\top}_{x,y} = (xy^2, xy, x) \tag{13}$$

Circle Fit for Glint

Regressing the glint is a subcase of regressing the ellipse in which one can choose the scaling a = b = 1, and has h = 0. Hence we have to fit the parameters $C = (c_x, c_y, r)$.

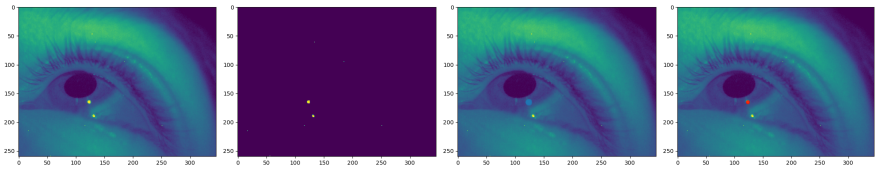
$$\begin{aligned} & E_{\mathscr{C}}(x,y) = 0 \\ \text{with} \quad & E_{\mathscr{C}}(x,y) = x^2 - 2xc_x - 2yc_y + (c_x^2 + c_y^2 - r^2) \end{aligned}$$

Frame Process

$$\mathscr{D}''_{\text{img.}} = \Big\{(x,y) \mid (x,y) \in H_{t_3}(I(x,y)) \text{ and } \|(x,y)-(x_e,y_e)\|^2 < \rho''\Big\}$$

Selected points are those that exceed a certain threshold t_3 and that are less than a certain distance ρ″ from the current pupil center fit.

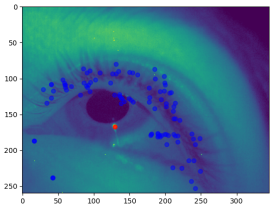
0. Original frames
1. Clip the pixels above some value
2. Select points (pixels) that are near the pupil center within some distance
3. Fit the circle



Event Process

$$\mathscr{D}''_{\text{evt.}} = \{(x,y) \mid |E_{\mathscr{C}}(x,y)| < \delta\}$$

Select event points by distance to the circle center.

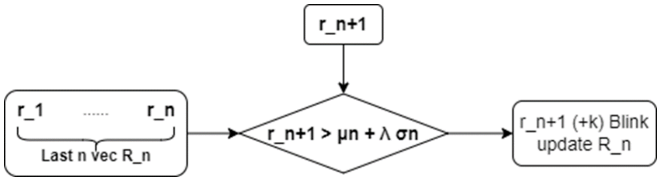


Circle Fit

Since it's a subcase of ellipse, its fit process is similar to the ellipse step.

Blink detector

Detect recent changes in eccentricity of pupil ellipse on frames. Once blink detected, the following several frames and events data will be ignored.



Gaze Map Regressor

Map from ellipse center (x_e, y_e) to screen pixel (x_s, y_s) by a polynomial regressor trained by the labels.

$$G_{\theta}(\mathscr{E}) = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} G_{\theta^1}|_x(\mathscr{E}) \\ G_{\theta^2}|_y(\mathscr{E}) \end{pmatrix}$$

$$G_{\theta^i}|_{x/y}(x_e, y_e) = \alpha_i x_e^2 + \gamma_i x_e y_e + \beta_i y_e^2 + \varepsilon_i x_e + \zeta_i y_e + \eta_i$$

$$\arg\min_{\theta^1} \|G_{\theta^1}|_x(x_e, y_e) - x_s\|^2$$
$$\arg\min_{\theta^2} \|G_{\theta^2}|_y(x_e, y_e) - y_s\|^2$$

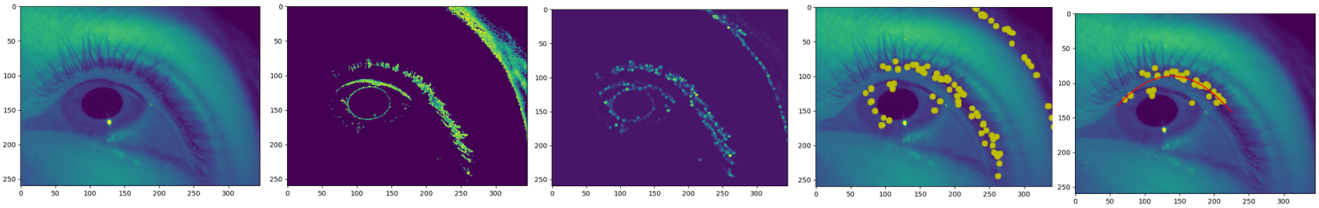
My Work of Improvement

We firstly reproduced these methods in paper. And we mainly did some robust improvement on fitting of pupil ellipse, eyelid parabola, and blink detection.

On Parabola of Eyelid Fitting

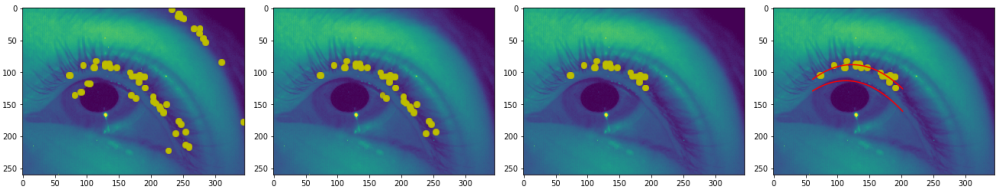
In the original paper method, when fit parabola of eyelid by event, on the corner point detection step, some points near the pupil is also detected. Since the situation not always happened, if we increase the taking percentage limit, when not the situation, there will be too less point to fit parabola well.

Bad Fit Example



When fit parabola of eyelid by event, we add a filter on corner points that check the distance to the pupil. Cut these corner points near enough to the pupil.

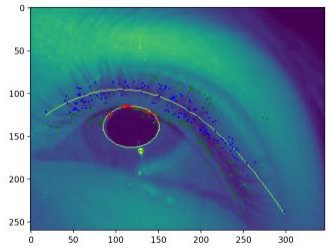
My Result



On Ellipse of Pupil Fitting

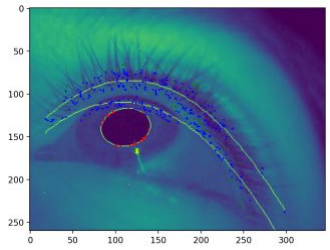
In the original paper method, when using event to fit pupil, it just use distance to the last fitted pupil to filter coming events. It may cause problem when some events are not caused by the pupil, such as lower bound of eyelid. These events will cause bad fit.

Bad Fit Example



In my method, we add second lower eyelid curve, share the parameters with main eyelid curve and add some offsets. The second eyelid helps further filter noise events for ellipse pupil fit. Before using event to fit the pupil, it will remove all the events near two eyelid curves.

My method

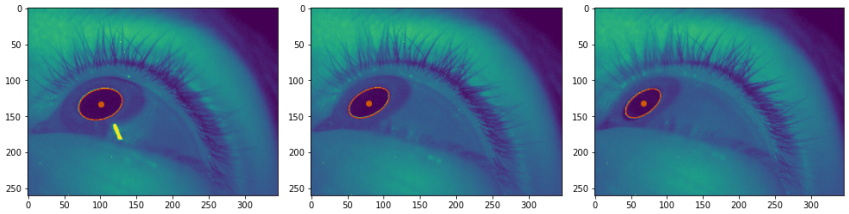


On Blink Detector

In the original paper method, it detect recent changes in eccentricity of pupil ellipse on frames, compare new fitted eccentricity with mean value add standard deviation. But it may misreport when the gaze angle changes quickly.

Bad Fit Example

3 adjacent frames, in which the eccentricity of the pupil ellipse changes quickly.



Thus, we changed the method to detect the change range of Parabola of Eyelid. If the parabola moves down far enough, it will report the blink.

On Circle of Glint Fitting

For the glint fitting method, the paper uses a circle to fit glint and see it as a subcase of ellipse. Considering the glint is very small, during fitting process there may be too few filtered points to success. Thus, we decided to calculate the average coordinates of the selected points as glint center, which can handle situation of few data or individual error data. It improved the stability of the glint fitting.

Add Kalman Filter

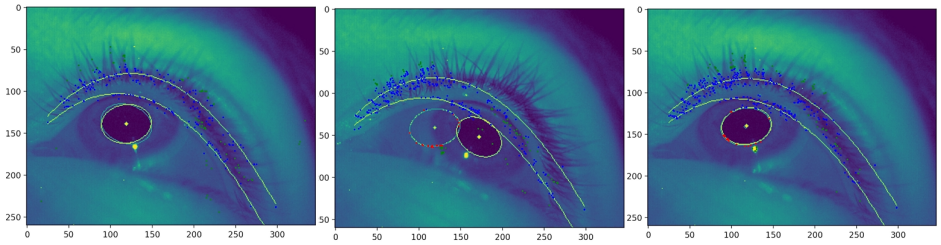
Since the movement of the pupil is continuous, we want to take use of the temporal information of the input data. We applied the Kalman filter to the pupil ellipse. So that the model can handle some situations like temporarily increased noise (external light change, etc.) by dropping some bad or failure model fits that are much different from the Kalman filter prediction.

We use 5 measurement parameters: ellipse's location (x, y pixel), size (short, long axis), and angle; and 10 state parameters: 5 states + change speed of 5 them. In the program, since the start state value can only be 0, we set all the parameters to the relative difference to the initial ellipse value.

$$state: x_k = [x_e \quad y_e \quad a \quad b \quad \theta \quad dx_e \quad dy_e \quad da \quad db \quad d\theta]$$
$$measurement: z_k = [x_e \quad y_e \quad a \quad b \quad \theta]$$

We use them to discard the bad or failed fits of the pupil ellipse. When the difference between the current fitted ellipse and the last prediction ellipse is exceed 10%, the current fit will be discarded and keep the prediction as formal fit. Other wise, we just use the fit normally to update the model and correct the Kalman filter. But the situation that kalman prediction is not correct sometimes happened.

3 adjacent frames (the middle frame is replaced on purpose). Green: Kalman, Yellow: current fit)



On Gaze Estimation

Gaze estimation means mapping the model data fitted from the eye images and event data to the screen to obtain the coordinates of the gaze point on the screen.

The paper's method use only pupil center coordinate as input. We considered to use more information that we extracted from the eye model to improve the mapping accuracy. We also proposed an alternative mapping method. By Jiahui [2] and Blignaut [3], using the pupil-glint 2D vectors is a typical 2D gaze estimation method. Using an appropriate polynomial model, the difference in x-coordinates (x') and difference in y-coordinates (y') between the pupil and glint can be mapped to screen coordinates (X,Y). Moreover, we also added the gaze vector to the model, which is calculated by the ellipse centrifugation.

Pupil-glint Vector

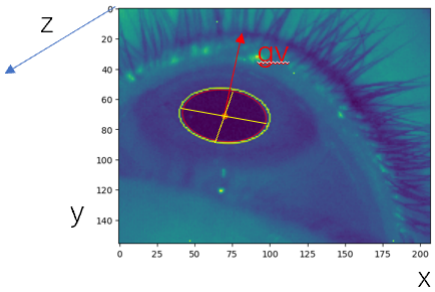
It's the difference in x-coordinates and difference in y-coordinates between the pupil center and glint.

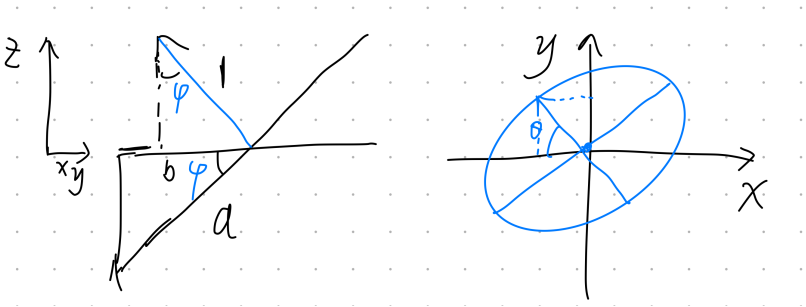
$$v_{pg} = (x_{pgv}, y_{pgv}) = \begin{bmatrix} x_p - x_g \\ y_p - y_g \end{bmatrix}$$

Gaze Vector

Suppose that the pupil is circular when looking directly at the pupil (at the center of the looking screen), the angle of offset can be calculated by fitting the long and short axes of the ellipse, and then obtaining the gaze unit vector.

Coordinate: xy plane is eye image and z vertical to xy.





Gaze unit vector: $v_g = (x_{gv}, y_{gv}, z_{gv}) = (\sin\varphi\cos\theta, \sin\varphi\sin\theta, \cos\varphi)$

Adding above elements to the gaze estimation model, we used a second order polynomial with 24 variables in total:

$$x_s = G_a(x_p, y_p)$$
$$= a_0 + a_1 x_p + a_2 y_p + a_3 x_p y_p + a_4 x_p^2 + a_5 y_p^2 + a_6 x_{gv}$$
$$+ a_7 y_{gv} + a_8 x_{gv} y_{gv} + a_9 x_{pgv} + a_{10} y_{pgv} + a_{11} x_{pgv} y_{pgv}$$

$$y_s = G_b(x_p, y_p)$$
$$= b_0 + b_1 x_p + b_2 y_p + b_3 x_p y_p + b_4 x_p^2 + b_5 y_p^2 + b_6 x_{gv}$$
$$+ b_7 y_{gv} + b_8 x_{gv} y_{gv} + b_9 x_{pgv} + b_{10} y_{pgv} + b_{11} x_{pgv} y_{pgv}$$

The parameters a, b are fitted by solving the following linear least squares

$$\arg\min_a ||G_a(x_p, y_p) - x_s||^2$$
$$\arg\min_b ||G_b(x_p, y_p) - y_s||^2$$

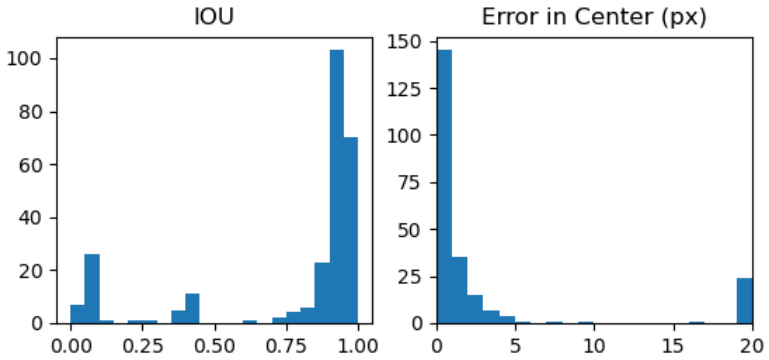
Pupil Tracking Evaluation

We evaluate the pupil tracking by the intersection over union (IOU) and error in ellipse center estimation. We compared the pupil fitted by a frame and the pupil fitted by events just before the frame.

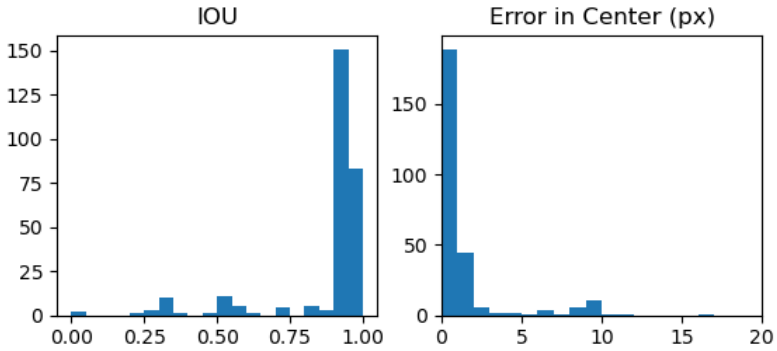
$$IOU(x, y) = \frac{\sum x \wedge y}{\sum x \vee y}$$
$$\text{center error} = ||(x_e, y_e) - (x_f, y_f)||$$

We tested the original and improved eye model models on a small dataset with 300 continuous frames and their event data. The result is shown below. The y-axis represents the count.

Paper method:



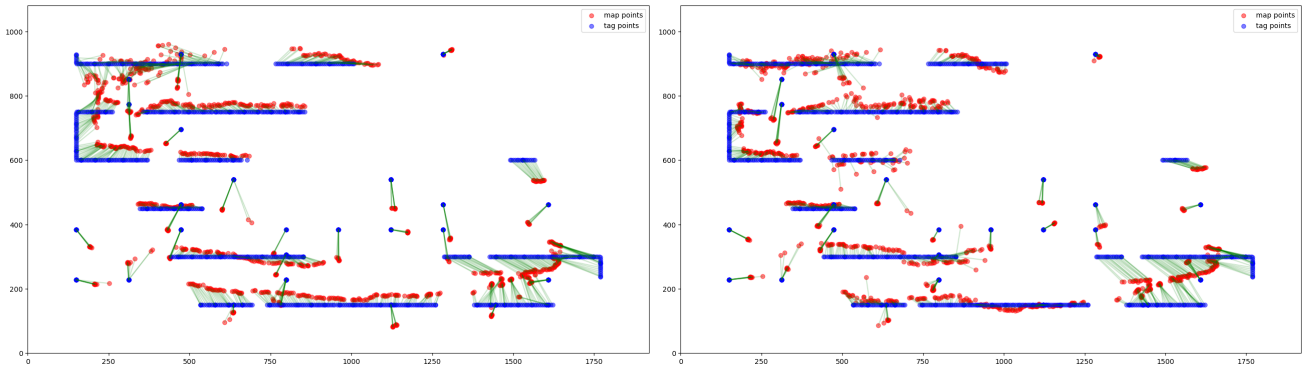
Our method:



In the paper method, we discarded many failure fits, so that its total valid count is less than ours. It means that our method has better robustness. We can see that we have both more good estimations and less bad estimations. Overall we have better distribution than the paper method.

Gaze Estimation Evaluation

We followed evaluation method in the paper. In gaze estimation evaluation, keeping the eye model same (both using our method), we trained original regressor and our regressor on same 400 frames and test on 1000 frames. Here's the visual test results, the blue represents points shown on screen, and the red represents gaze estimated points.



Paper method Our method

We calculated the average distance between tag point coordinates and estimated point coordinates. Paper method gives $\bar{d}_p = 60.428$ and our method $\bar{d}_m = 48.516$. We have better performance.

$$\bar{d} = \frac{1}{n} \sum ||(x_{est}, y_{est}) - (x_{tag}, y_{tag})||_2$$

Furthermore, we calculated the accuracy and precision methods. By the paper, “accuracy” results are calculated using the ISO 5725 definition of “trueness”: the closeness of agreement between the arithmetic mean of a large number of test results and the true or accepted reference value, this is

$$\text{Accuracy} = \frac{1}{n} \sum_{i=1}^{n} l_i \tag{24}$$

where n is the total number of estimates, and l_i is the L_2 norm of the difference between our estimated gaze direction $\hat{d}_i = (\hat{\phi}_i, \hat{\theta}_i)$ and the true gaze angle $d_i = (\phi_i, \theta_i)$:

$$l_i = \|\hat{d}_i - d_i\|_2 \tag{25}$$

Precision is defined by ISO 5725 as: “the closeness of agreement between test results”. We quantify this by computing the empirical standard deviation of the $\hat{d}_i$’s:

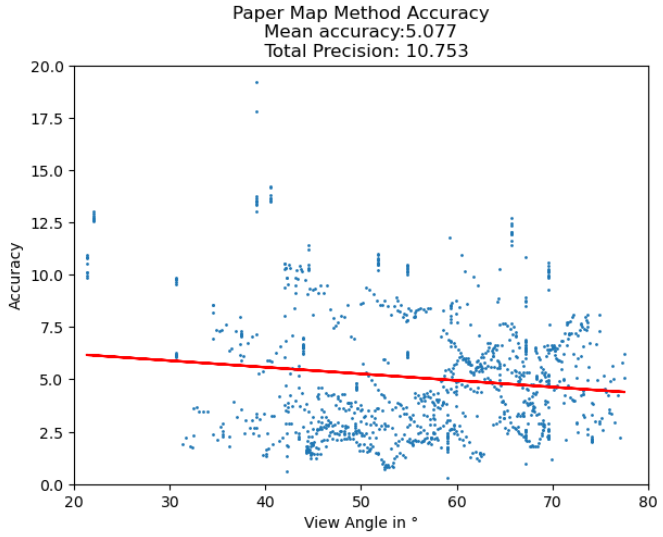
$$\text{Precision} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\hat{d}_i - \bar{\hat{d}})^2} \qquad (26)$$

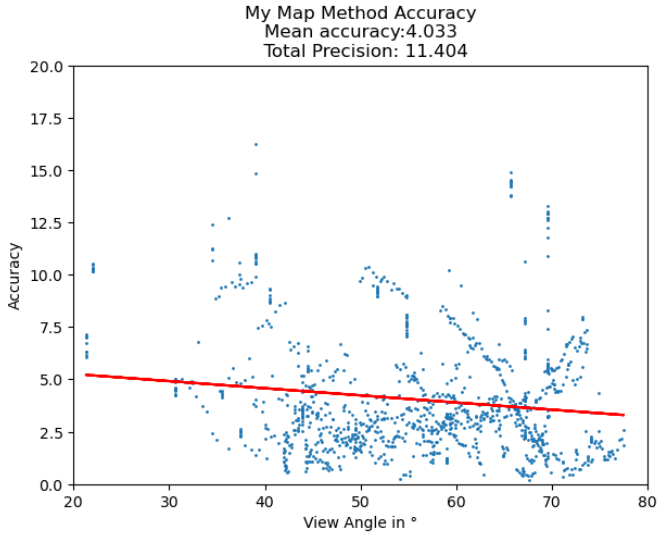
The relationship between the horizontal angle θ , the vertical angle φ , and the screen coordinates which a user looks at (x_s, y_s), is trigonometric because the screen is at a fixed distance. With (c_x, c_y) being the center of the screen at a fixed distance D, The conversion is calculated simply as:

$$\theta = \frac{180}{\pi}\tan^{-1}(|x_s - c_x|/D)$$
$$\phi = \frac{180}{\pi}\tan^{-1}(|y_s - c_y|/D)$$

Paper method gives mean accuracy 5.077, precision 10.753, and our method's mean accuracy 4.033, precision 11.404. Our method improved the accuracy about 20%, but the precision decreased about 6%. Overall, our method have better performance.

We also explored the variation of accuracy with viewing angle and compared them. Although the regression line was drawn here, but not that much credibility.





Summary

In this research project, I mainly reproduced and improved the event based eye tracking method by Angelopoulos [1]. I did some robust improvement on fitting of eye model and accuracy improvement on gaze estimation method. But finally it's still using frame and event mixed data as input. The model and method that can handle pure event data is desired. I specially appreciate Professor Qilin Sun for his guidance on my project.

Reference

[1] Angelopoulos, *Event Based, Near Eye Gaze Tracking Beyond 10,000 Hz*, IEEE Transactions on Visualization and Computer Graphics, 2021,27(5): 2577 - 2586

[2] Jiahui Liu, Jiannan Chi, Huijie Yang, Xucheng Yin, *In the eye of the beholder: A survey of gaze tracking techniques*, Pattern Recognition, 2022, 132: 108944

[3] Pieter Blignaut, *Mapping the Pupil-Glint Vector to Gaze Coordinates in a Simple Video-Based Eye Tracker*, Journal of Eye Movement Research, 2013, 7(1):1-11

To Run My Code

run whole steps:

run python model_test.py

run single steps:

run model_test.ipynb

view dataset

run python visualize.py