S E S   2 0 2 0
Sixteenth International Scientific Conference
SPACE, ECOLOGY, SAFETY
4 – 6 November 2020, Sofia, Bulgaria

A TOOL FOR ESTIMATING SACCADE KINEMATICS

Konstantin Metodiev

Space Research and Technology Institute – Bulgarian Academy of Sciences
e-mail: komet@space.bas.bg

Keywords: *eye tracker, pupil, OpenCV, saccade, fixation*

Abstract: *In the paper hereby, an algorithm is proposed for estimating amplitude and peak velocity of a saccade during directing eyes towards next fixation. Raw data are gathered by means of a mobile eye tracker. The non-linear least squares fitting is applied so as to minimize difference between measured and modelled values of a gaze point. A sigmoid function has been used to fit the data set best. In order to implement the algorithm, a software tool has been developed in C++ making the most of linear algebra methods available in OpenCV library.*

СРЕДСТВО ЗА ОЦЕНКА КИНЕМАТИКАТА НА САКАДА

Константин Методиев

Институт за космически изследвания и технологии – Българска академия на науките
e-mail: komet@space.bas.bg

Ключови думи: *окулограф, зеница, OpenCV, сакада, фискация*

Резюме: *В настоящия доклад е разгледан алгоритъм за оценка на амплитудата и пиковата скорост на сакада при движение на погледа към последваща фиксация. Данните са измерени посредством мобилен окулограф. Приложен е нелинеен метод на най-малките квадрати за минимизация на разликите между измерената и моделирана стойности на точката на взиране. Използвана е сигмоидна функция, която минава най-близо до измерените точки. За реализация на алгоритъма бе разработен софтуер на C++ с използване на методи по линейна алгебра, налични в библиотека OpenCV.*

Introduction

A saccade is recognized as the fastest motion that human body performs. It is a swift, lacking a steady rhythm movement of the eye ball performed between pauses for relaxation. The relaxation phase is commonly referred to as fixation. The act of sight directing along the saccade trajectory is expressed numerically by an angular variable called amplitude. The greatest rate of angular movement is denoted as peak velocity. As acknowledged, the saccade velocity cannot be altered through either practice or experience. Furthermore, a condition might be reached at which saccade peak velocity gets to a limit. The limit value (a.k.a. velocity saturation) is a distinctive attribute of large saccades.

Aforementioned quantities might be derived through infrared oculography. It is a non-invasive method of recording eye movement using an infrared emitter and a detector. The amount of reflected infrared light is proportional to the pupil position. This method has been employed by many eye tracker manufacturers, for instance Pupil Labs, and further implemented in supplied firmware.

Although Pupil Player makes available a number of data processing plug-ins, it lacks a tool for extracting saccades from raw data set. Making up for this shortcoming is one reason motivating to carry out current research. An attainable and feasible outcome is developing a software tool, yielding robust results, for estimating saccade amplitude and peak velocity. The widely known library of image processing OpenCV, [1], is quite useful for this purpose. C++ API provides methods in linear algebra as well as tools for image processing. Code development was thus facilitated to a certain extent.

The paper lays out project main stages in a sequence of cohesive paragraphs.

Problem statement

The eye tracker used in the current study is Pupil Labs Core, [2], Fig. 1.

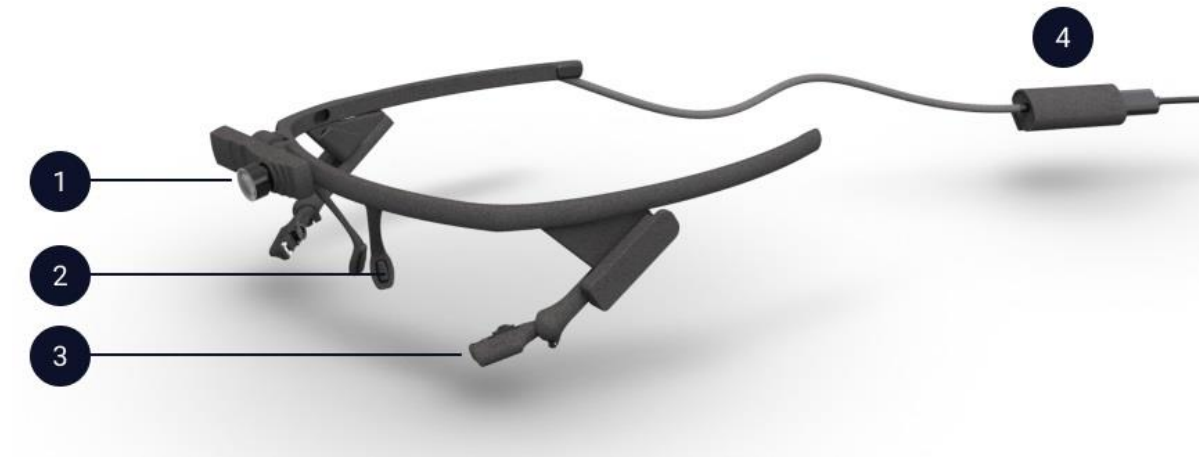
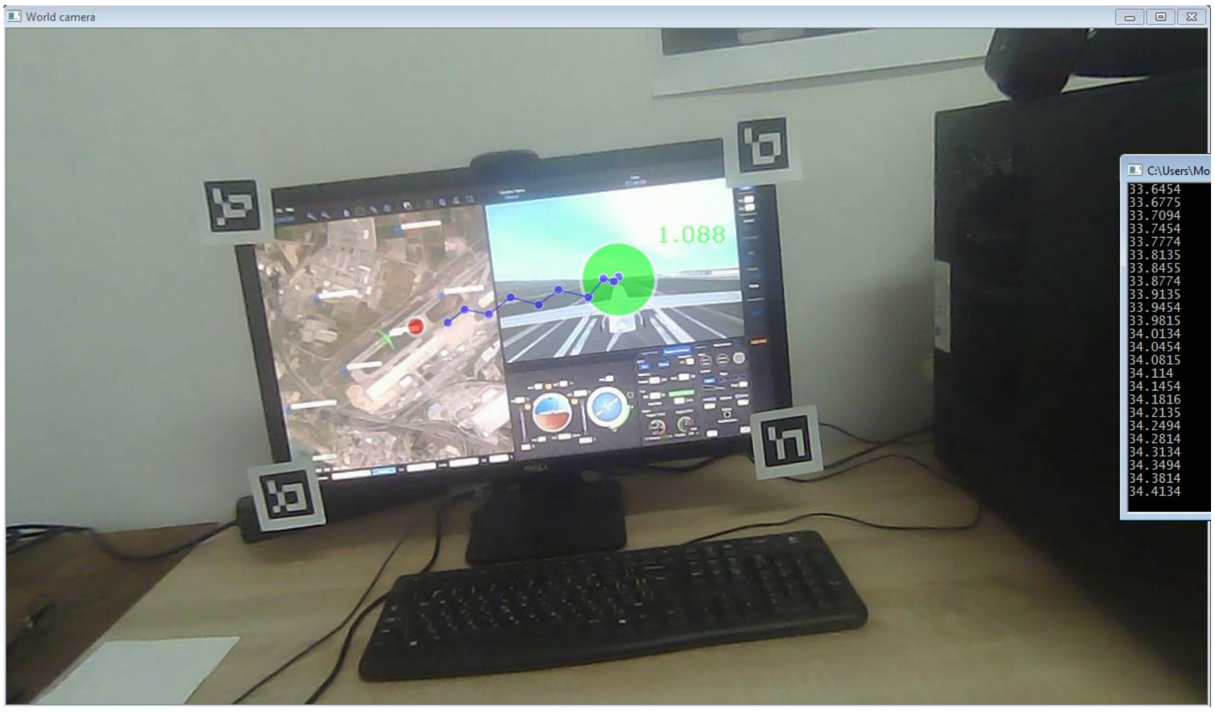


Fig. 1. Pupil Labs Core binocular headset: 1) world camera; 2) nose grip; 3) near infrared camera; 4) USB-C interface

A screenshot, taken by Pupil world camera and shown in Fig. 2, depicts the proposed study case. On the screen, two fixations appear in succession with radii computed with regard to fixation duration. The green circle depicts a fixation in progress whereas the red one portrays a fixation that follows. Looking into intermediate gaze points (small blue circles), which appear right between aforementioned fixations, is a matter of particular importance. These points are rejected by the algorithm of fixation identification. The entire set of gaze points appears like a saccade.



```
33.6454
33.6775
33.7094
33.7454
33.7774
33.8135
33.8455
33.8774
33.9135
33.9454
33.9815
34.0134
34.0454
34.0815
34.114
34.1454
34.1816
34.2135
34.2494
34.2814
34.3134
34.3494
34.3814
34.4134
```

Fig. 2. A screenshot depicting two consecutive fixations and intermediate gaze points

The dispersion-threshold identification algorithm (I-DT), [3], is said to have been implemented by Pupil Player in order to identify fixation. It uses a moving window spanning over sequential gaze points. The dispersion is computed for all points within the window as follows:

(1) $$D = \max(x) - \min(x) + \max(y) - \min(y)$$

The window keeps on widening till requirement *0 < D ≤ threshold* is being met. As soon as dispersion exceeds the assigned threshold, a fixation is recorded. After resetting size, the window advances as toward next gaze point (following fixation) and the process repeats. In this way, certain gaze points are shown not to belong to fixation on either side of what is allegedly called a saccade.

Nonlinear Least Squares Fitting

The measured saccade points (x_i, y_i) presumably contain error. Hence, an analytic function f(x, λ_j) depending on n parameters λ is to be found so as to minimize the residual

(2) $$d\beta_i = y_i - f\left(x_i, \lambda_1, \ldots, \lambda_n\right)$$

The solution scheme used in the current study is widely known. It borrows an algorithm thoroughly described in link [4]. Having made initial guess for a set of coefficients $\lambda_j^{(1)}$, a linearized estimate of disturbances $d\lambda_j$ minimizing $d\beta_i$ may be obtained by means of following series

(3) $$d\beta_i = \sum_{j=1}^{n} \frac{\partial f}{\partial \lambda_j} d\lambda_j \bigg|_{x_i, \lambda_j^{(1)}}$$

where index i ϵ [1; m] stands for the current measurement and j ϵ [1; n] denotes current coefficient vector. Equation (3) might be written down by means of matrix notation

(4) $$\begin{Vmatrix} d\beta_1 \\ d\beta_2 \\ \vdots \\ d\beta_m \end{Vmatrix} = \begin{Vmatrix} \frac{\partial f\left(x_1, \lambda_j^{(1)}\right)}{\partial \lambda_1^{(1)}} & \frac{\partial f\left(x_1, \lambda_j^{(1)}\right)}{\partial \lambda_2^{(1)}} & \cdots & \frac{\partial f\left(x_1, \lambda_j^{(1)}\right)}{\partial \lambda_n^{(1)}} \\ \frac{\partial f\left(x_2, \lambda_j^{(1)}\right)}{\partial \lambda_1^{(1)}} & \frac{\partial f\left(x_2, \lambda_j^{(1)}\right)}{\partial \lambda_2^{(1)}} & \cdots & \frac{\partial f\left(x_2, \lambda_j^{(1)}\right)}{\partial \lambda_n^{(1)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f\left(x_m, \lambda_j^{(1)}\right)}{\partial \lambda_1^{(1)}} & \frac{\partial f\left(x_m, \lambda_j^{(1)}\right)}{\partial \lambda_2^{(1)}} & \cdots & \frac{\partial f\left(x_m, \lambda_j^{(1)}\right)}{\partial \lambda_n^{(1)}} \end{Vmatrix} \square \begin{Vmatrix} d\lambda_1 \\ d\lambda_2 \\ \vdots \\ d\lambda_n \end{Vmatrix}$$

or simply

(5) $$d\beta_i = A_{i,j} d\lambda_j$$

Solution of equation (5) yields a disturbance vector $d\lambda_j$ value which must be added to vector $\lambda_j^{(prev)}$ available at previous iteration. Then, a new residual $d\beta_i$ is computed according to formula (2) and the process starts over. For the matrix of coefficients to get invertible, its transpose is applied to both sides of (5), id est:

(6) $$A_{j,i}^T d\beta_i = \left(A_{j,i}^T A_{i,j}\right) d\lambda_j$$

Provided that det($A_{j,i}^T A_{i,j}$) ≠ 0, for the solution vector it may be written

(7) $$\left(A_{j,i}^T A_{i,j}\right)^{-1} A_{j,i}^T d\beta_i = I_{j,j} d\lambda_j$$

where $I_{j,j}$ stands for identity matrix.

Numerical solution computed by described iterative process is likely to converge to a local extremum than the global one, [5], depending on whether the model function is sensitive to initial conditions or not. In case of a poorly performing system, it is reasonable for the initial conditions to influence the solution convergence. Proper selection of "good" initial conditions $\lambda_j^{(1)}$ is essential for the model function f(x, λ_j) to be "good of fit". A simple method of making initial guess might be devising a grid of values for each coefficient taking part in the model function. Then, for each set of initial values of coefficients, a solution is to be worked out followed by computing the residual standard deviation (RSD) according to formula:

$$RSD = \sqrt{\frac{\sum_{i=1}^{m} d\beta_i^2}{m-2}} \tag{8}$$

Vector of initial values $\lambda_j^{(1)}$ corresponding to the smallest RSD is to be chosen for initial guess.

Among variety of sigmoid functions applicable to the current study, it is rather preferable to choose one with as few coefficients as possible. To opt for a model function with large number of coefficients may result in poor algorithm performance in terms of convergence speed. What is more, numerical solution in this case is highly likely not to converge at all. In the current research, a sigmoid function with three coefficients is considered sufficient for the purpose.

A short note of developed source code

A solution to equation (7) is worked out by means of linear algebra methods available in OpenCV library. It takes the disturbance vector dλ_j exactly one line of code to move on further to the next iteration step:

```
lambda += (a.t() * a).inv() * a.t() * beta; // delta lambda plus lambda
```

Methods ::inv() and ::t() provide a convenient way for developer to compute matrix inverse and transpose respectively.

Saccade model

A logistic function has been used to fit the measured gaze points between two consecutive fixations. It could be written down in following form:

$$\theta(t) = \frac{a}{1+e^{\frac{t_0-t}{b}}} \tag{9}$$

where scaling coefficients modify the graph as follows: coefficient "t_0" shifts the graph along abscissa, coefficient "b" affects tangent angle at the inflection point (scale along abscissa), and coefficient "a" scales the graph along ordinate. In current study, shifting graph along the ordinate is considered unnecessary, hence no relevant coefficient has been added. Exemplary logistic curve is shown in Fig. 3 alongside its first derivative (dashed curve, right ordinate).

Following derivatives in terms of model function coefficients λ_j are used in order to fill in the matrix in equation (4)

$$\frac{\partial\theta}{\partial a} = \frac{1}{1+e^{\frac{t_0-t}{b}}};\quad \frac{\partial\theta}{\partial t_0} = -\frac{a}{b}e^{\frac{t_0-t}{b}}\frac{1}{\left(1+e^{\frac{t_0-t}{b}}\right)^2};\quad \frac{\partial\theta}{\partial b} = \frac{a}{b^2}(t_0-t)e^{\frac{t_0-t}{b}}\frac{1}{\left(1+e^{\frac{t_0-t}{b}}\right)^2} \tag{10}$$

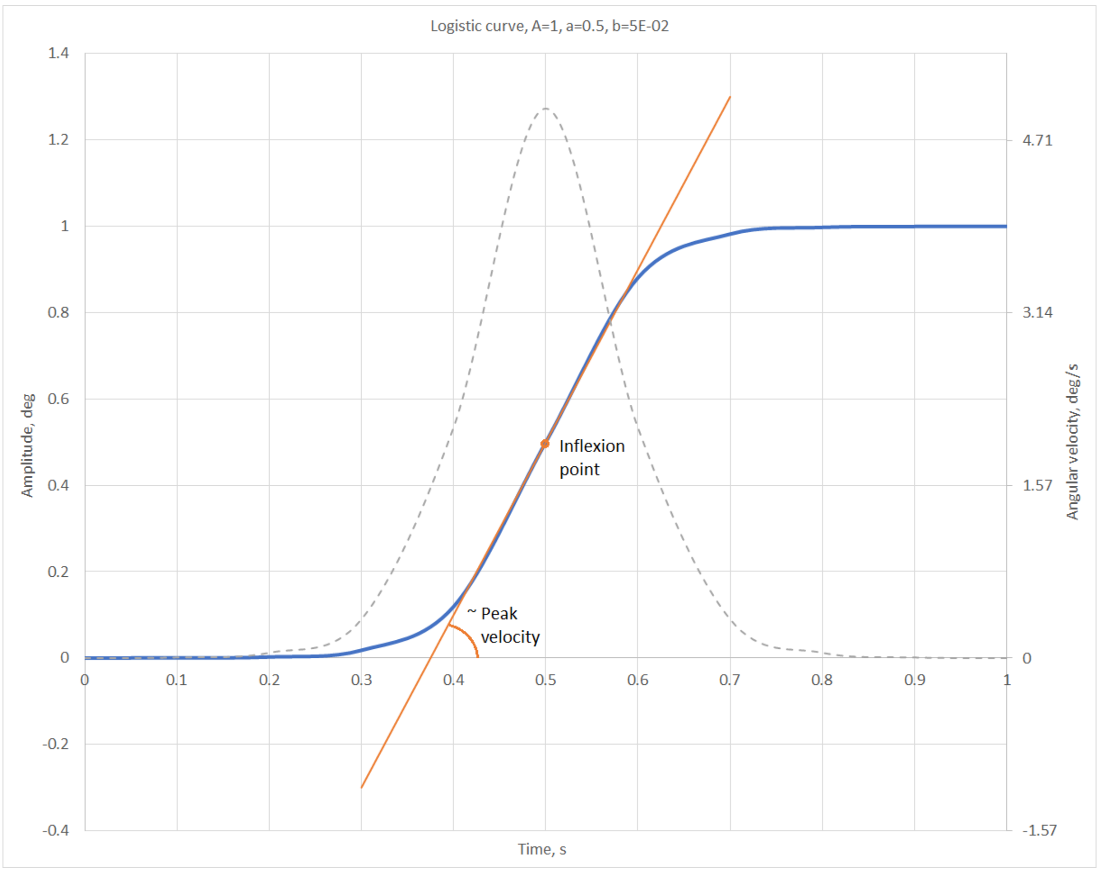


Fig. 3. A sigmoid function (logistic curve) used in the study

Results and discussion

In order to work out a solution to equation (7) for validation case shown in Fig. 2, following table was filled in with initial values of model function coefficients:

Table 1. Grid of initial conditions

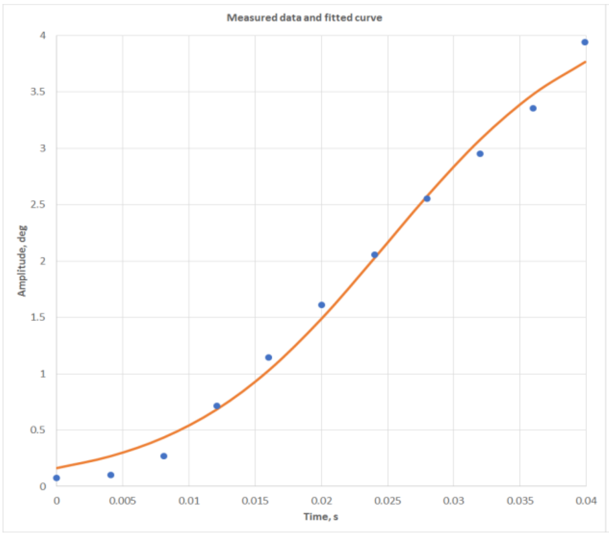
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
λ_0 = a	3.6	3.7	3.8	3.9	4	4.1	4.2	4.3
λ_1 = t_0	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07
λ_2 = b	0	0.005	0.01	0.015	0.02	0.025	0.03	0.035

Considering table 1, distinct values of three coefficients are picked out of eight columns. Identical indices are permitted, for instance row 2, column 2. Since the order does matter, this is an example of a variation with repetitions allowed. The number of 3-element (rows) variations of 8 elements (columns) set with repetitions allowed is 512. The solution algorithm is run for each variation and obtained RSD values are compared. It takes one solution case approximately 15 iterations to converge. The iterative process stops as soon as following termination criterion is satisfied at kth step

(11) $$\left\|\mathbf{x}^{(k)} - \mathbf{x}\right\| \le \varepsilon$$

where **x** is an exact solution vector and ||.|| stands for a vector norm.

In case of initial values found in cells (λ_0; 4), (λ_1; 2), (λ_2; 1), table 1, shaded cells, the RSD takes up the smallest value of 0.13178. Solution vector of the model function coefficients is λ = $|4.300159809398955;\ 0.0248915584037846;\ 0.00766740851114877 8|^T$. Another choice of initial coefficients (λ_0; 0), (λ_1; 3), (λ_2; 2) yields much about the same value of RSD, id est 0.13178. Both measured saccade amplitude and fitted logistic curve are shown in Fig. 4a. Distance between user's eye and the monitor is assumed 80 cm, hence the eye ball rotates within [0; 4] deg interval.



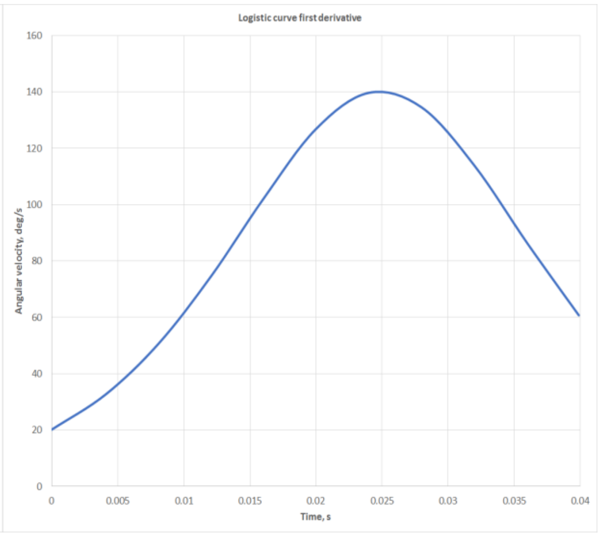


Fig. 4a. Measured data and fitted curve

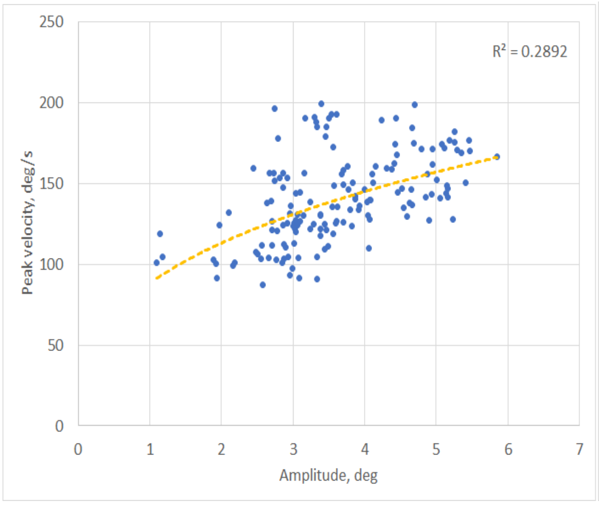
Fig 4b. First derivative of the fitted curve

Having obtained the logistic curve coefficients, it is possible to work out the peak velocity value by computing first derivative of the model function according to expression:

(12) $$\frac{d\theta}{dt} = \frac{a}{b} e^{\frac{t_0 - t}{b}} \left(1 + e^{\frac{t_0 - t}{b}}\right)^{-2}$$

A function plot of derivative dθ/dt is shown in Fig. 4b. Peak velocity value for this particular case is 140.202 deg/s achieved on the instant of 25 ms.

In Fig. 5a, a relation between saccade peak velocity and saccade amplitude is shown, so is saccade duration in terms of saccade amplitude, Fig. 5b. The experiment lasted for 150 seconds. All the way through the experiment, near infrared cameras were set to gather data at resolution of 800 x 600 pixels, hence the frequency dropped as low as 30 Hz. In Fig. 5a, weak correlation between input and output data might be accounted for by a small range within which the amplitude varies. Saturation of peak velocity value is solely expected at high saccade amplitudes.



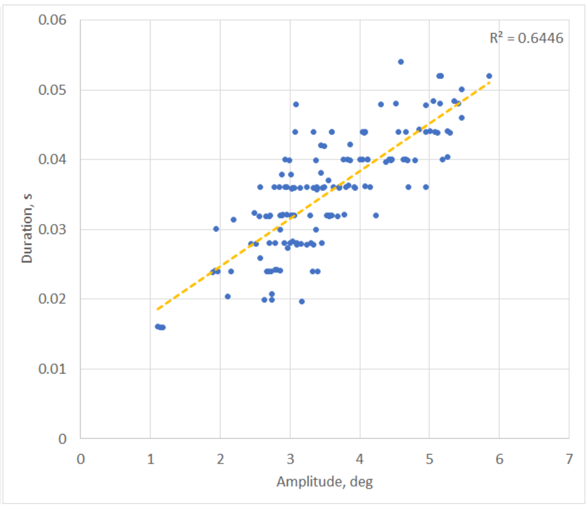


Fig. 5a. Peak velocity, deg/s, vs. saccade amplitude, deg

Fig 5b. Duration, s, vs. saccade amplitude, deg

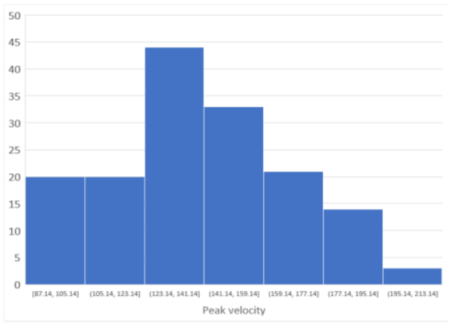


Fig. 6a. Peak velocity histogram, bin width 18 deg/s

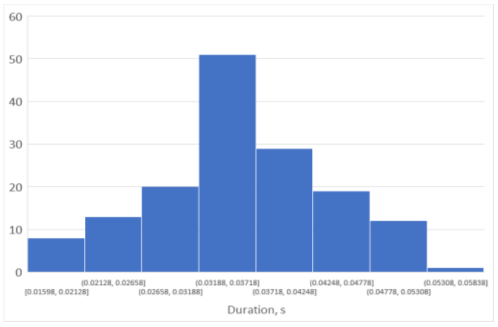


Fig 6b. Duration histogram, bin width 5.3E–03 s

Concluding remarks

In order to fulfil the project goals, following software was used: TDM64-GCC-9.2.0, OpenCV 4.4.0. The OpenCV project was configured by CMake 3.18.2 and binaries were built in Code::Blocks 20.03 IDE, [6 … 9]. A comprehensive tutorial on how to build the OpenCV binaries might be found in [10]. Apart from all configuration options described in the tutorial, following one is worth pointing out: WITH_TBB = ON which stands for enabling multicore CPU support.

The developed source code might be downloaded by following link [11].

Acknowledgements

This research has been supported by Bulgarian National Science Fund at the Republic of Bulgaria according to contract № КП-06/Н27-10, 11th of December, 2018. The project title is “Human Factor in Remotely Controlled Aerial Systems – Analysis, Estimation, and Control.”

References:

1. Bradski, G., The OpenCV Library, Dr. Dobb's Journal of Software Tools, article id=2236121, 2000
2. Kassner, M., W. Patera, A. Bulling, Pupil: An Open Source Platform for Pervasive Eye Tracking and Mobile Gaze-based Interaction, arXiv:1405.0006v1 [cs.CV] 30 Apr 2014
3. Salvucci, D., J. Goldberg, Identifying Fixations and Saccades in Eye-Tracking Protocols, Proceedings of the Eye Tracking Research and Application Symposium, ETRA 2000, Palm Beach Gardens, Florida, USA, November 6-8, 2000
4. https://mathworld.wolfram.com/NonlinearLeastSquaresFitting.html
5. https://www.itl.nist.gov/div898/handbook/pmd/section6/pmd632.htm
6. https://jmeubank.github.io/tdm-gcc/
7. http://www.codeblocks.org/
8. https://opencv.org/releases/
9. https://cmake.org/
10. https://medium.com/@sourabhjigjinni/install-opencv-4-0-0-for-c-windows-7-10-code-blocks-tdm-gcc-64-dff65addf162
11. https://github.com/samolet4e