Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 685–694

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Deep Neural Networks for Low-Cost Eye Tracking

Ildar Rakhmatulin[a], Andrew T. Duchowski[b*]

[a]South Ural State University, Department of Power Plants Networks and Systems, Chelyabinsk, 454080, Russia
[b]Clemson University, 100 McAdams Hall, Clemson, SC 29634, USA

Abstract

The paper presents a detailed analysis of modern techniques that can be used to track gaze with a webcam. We present a practical implementation of the most popular methods for tracking gaze. Various models of deep neural networks that can be involved in the process of online gaze monitoring are reviewed. We introduce a new eye-tracking approach where the effectiveness of using a deep learning method is significantly increased. Implementation is in Python where its application is demonstrated by controlling interaction with the computer. Specifically, a dual coordinate system is given for controlling the computer with the help of a gaze. The first set of coordinates–the position of the face relative to the computer, is implemented by detecting color from the infrared LED via the OpenCV library. The second set of coordinates–giving gaze position–is obtained via the YOLO (v3) package. A method of labeling the eyes is given, in which 3 objects are used to track gaze (to the left, to the right, and in the center).

Keywords: eye tracking; deep learning in eye tracking; yolov3 in eye tracking; deep learning in gaze tracking

1. Introduction

Widespread adoption of eye-tracking technology has expanded its application. For example, eye-tracking systems are now used to detect disorders associated with multiple psychological illnesses [13]. Maurage et al. [18] gave a detailed account of using an eye-tracking system to determine alcoholism. Bueno et al. [2] submitted a dataset on the methods of eye tracking in neurodegenerative conditions and its potential clinical effect on cognitive assessment. Robertson et al. [23] showed that eye tracking reveals subtle problems in understanding speech in children with dyslexia. Eye tracking is also widely used to improve learning. Sun et al. [24] used an eye-tracking device to show the effectiveness of teaching a programming course in the C language. An empirical analysis based on eye tracking and subjective perception of students was described by Molina et al. [19]. Recently, using eye tracking has become popular in determining the physical condition of a person. Li et al. [16] used an eye-tracking

* *E-mail address:* ildar.o2010@yandex.ru (Ildar Rakhmatulin), duchowski@clemson.edu (Andrew T. Duchowski)

1877-0509
10.1016/j.procs.2020.09.041

system for the identification and classification of mental fatigue of construction equipment operators.

These examples show that despite the variety of areas in which eye-tracking systems are used, often low-cost tools with low tracking accuracy are used to implement the technology. This suggests a need for low-cost eye-tracking devices comparable in quality to laboratory equipment, which would allow scientists from various fields to obtain results with high accuracy. It is also important that the source code be open, allowing customization of the software. The main limitation of modern eye-tracking systems is their price. Equipment with an accuracy of about $1-2^{\circ}$ costs several thousand dollars. Here, we develop a low-cost eye-tracking system based on deep learning methods.

1.1. Overview of Modern Eye-Tracking Methods

Today, a particularly popular type of eye tracker is of the remote variety. With these trackers, the eyes are illuminated by direct invisible Infra-Red (IR) light, which leads to the appearance of its reflection on the cornea, which the camera tracks. The physiology of this process is described in detail by Hari [5]. In this paper, we use a similar method, where an infrared LED is used to determine the position of the eye relative to the computer monitor. Huang et al. [6] used a recurrence module to refine the orientation of the eyes using the initial shape of the eyes. The proposed algorithm can effectively supplement training data. The implementation of heat maps presented in this paper was used in this study to verify the operation of a developed tracking device. This method does not give an accurate assessment of the performance of the eye-tracking device and can only be used only for demonstration purposes. A new gaze detection model with a combination of superpixel segmentation and eye-tracking data was proposed by Fen et al. [17]. This study used a high-resolution image of the eyes. Such an image cannot be obtained from a standard webcam. Fen et al. did not describe how to focus the camera on the eyes. Ozcelik et al. [20] explained that the effect of color coding affords more accurate eye movement. In our work we use color-coding for the labeling process (while shooting, the eye looks at colored objects). Kerr et al. [11] presented a real-time correction method for eye tracking. This method implements the analysis of the collected calibration data from the user using the nearest neighbor calibration points to calculate the predicted drift at the current point of focus. Kerr et al. provide empirical data from participants suffering from Amblyopia. Larsson et al. [14] presented the development of a method that reliably detects events in signals recorded using a mobile eye-tracking device. The proposed method compensates for head movements recorded using an inertial measuring unit. In our research an infrared LED showed greater accuracy in compensating for head movements. Huang et al. [7] proposed a two-phase CNN learning strategy for combining head postures and viewing angles. The CNN architecture can reduce refit while training eye-tracking models directly with a head pose. In our research, the pose of the head and its effect on the eyes did not increase the tracking accuracy of the developed eye-tracking system. Jagtap et al. [8] implemented a method for monitoring driver fatigue to prevent traffic accidents. To determine the facial features of the driver, the Viola-Jones Classifier was used. In our work, due to the low tracking efficiency of small objects, the Viola-Jones Classifier was used only to determine the position of the head. Pavlas et al. [21] provided practical guidance on the creation of low-cost tracking. They used EyeWriter and ITU GazeTracker software in their system. The price of the device was about 100 dollars with an accuracy of 2 degrees. Although our device has good characteristics, it does not have the same versatility and is not quite as convenient to use. Lee et al. [15] proposed a method for assessing the position of three-dimensional gaze, based on illumination reflections (Purkinje images) on the surface of the cornea and lens, taking into account the three-dimensional optical structure of a model of the human eye. 3D modeling of the eye is a promising direction, but this requires high-quality images. This approach presents a promising direction for eye tracking because they use the 3D plane for analyzing an eye position. To resolve Purkinje images on the eye requires a high-resolution camera. For eye-tracking, we use a web camera, where 3D modeling is difficult to implement. Several researchers described the use of slow deep neural networks for image processing, such as Fast R-CNN, Faster R-CNN. For medical purposes, image processing can occur offline, but for real-time use, as in this work, stream mode is required. Many eye-tracking systems carry certain limitations, such as head position, lighting in the room. It is worth considering work in which the task was to develop a low-cost eye tracker. Javier et al. [9] presented a low-cost gaze tracking system that is based on a webcam mounted close to the user's eye, but accuracy of tracking was low. Frank et al. [4] introduced a system of stroboscopic catadioptric eye tracking (SCET). The new approach for mobile eye tracking, based on cameras with roller shutters and stroboscopic structured infrared lighting was described. This method has high accuracy but requires a complicated setup and can easily go astray if the computer tracks glare. Swarts et al. [25] presented an ultra-low-cost eye gaze tracker

specifically aimed at studying visual attention in 3D virtual environments. The authors do not provide conclusive evidence regarding accuracy of the developed device. Davin et al. [3] submitted information on how to build a low-cost eye-tracking system, work particularly interesting from the perspective of image processing. The closest prior art to our work is that of Krafka et al. [12], who presented software GazeCapture. GazeCapture, which contains data from more than 1,450 people, consists of almost 2.5 million pictures of people. In our work, we use similar technology for processing the face: finding the area with the eyes and using the convolutional neural network to track the eyes in this limited area. The difference in our work is a new labeling technique (using the eye and the inner eye gap for training the neural network) and the use of an infrared LED.

2. Image Preprocessing

The first step towards building a model for the tracking task is image preprocessing. Image pre-processing provides visualization necessary to identify and evaluate the usefulness of the procedure. Fig. 1 shows the implementation of the most popular pre-processing functions of the OpenCV library on the eye image with a resolution of 416×416 pixels.

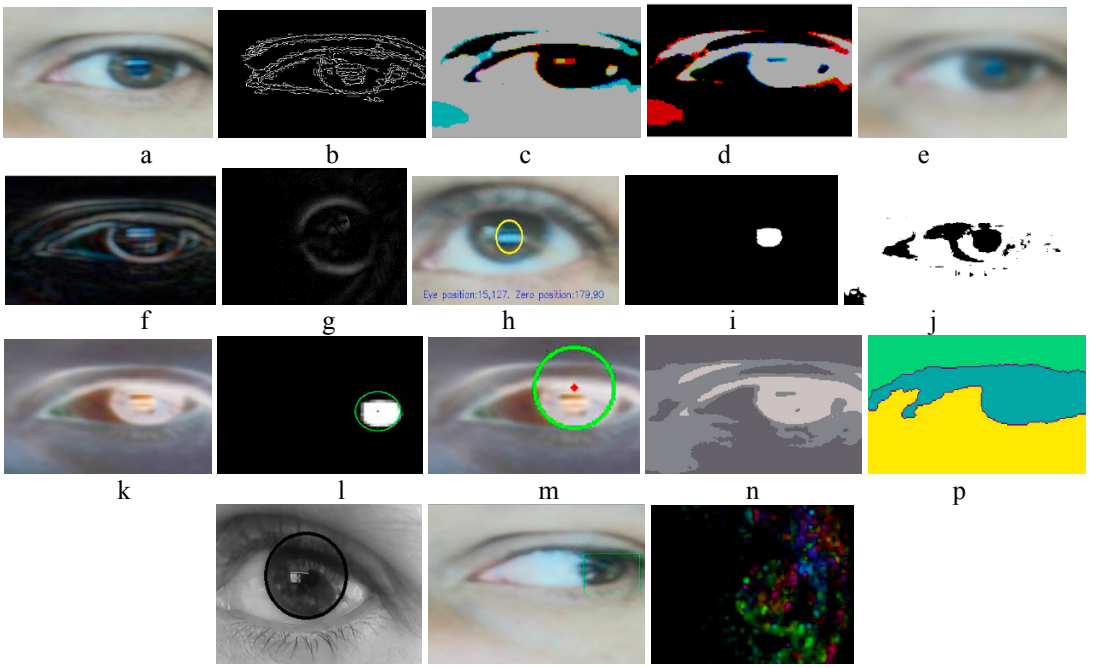


Fig. 1. Image pre-processing: a – source image, b – edge detection (cv2.Canny), c – threshold operation (cv2.threshold), d – threshold binary (cv2.threshold), e – Gaussian filter (cv2.GaussianBlur, f – morphology filter (cv2.morphologyEx), g – frame difference (cv2.absdiff), h – mask for detect color (cv2.inRange), i,j – for find colorLower, colorUpper characteristic (cv2.inRange), k – image segmentation, l,m – (cv2.HoughCircles), n – K-Means Clustering for Image Segmentation (cv2.kmeans), p – Image Segmentation with Watershed Algorithm (cv2.watershed), q – circles (cv2.HoughCircle), r – tracker (cv2.TrackerCSRT), s – flow (cv2.calcOpticalFlowFarneback)

The methods that process the color of the image are the most informative. Color, however, is not constant, therefore, these methods are not highly efficient.

3. Methods for Finding Eyes in the Video Stream

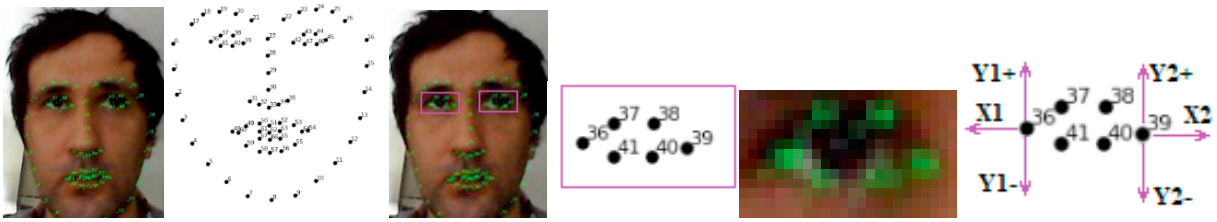
The use of Haar cascades and the dlib library directly to monitor the gaze position showed extremely low and unstable results. But at the same time, these libraries can be used as auxiliary methods for finding the eye region.

Fig. 2 shows the Haar cascade and dlib implementations.

a b c d

Fig.2. Eye position search: a – haar cascade, b,c,d – various dlib library implementation

The Haar cascade is a machine-learning method for detecting objects in an image, the idea of which was proposed by Viola and Jones [26]. A trained Haar cascade, taking an image as input, determines whether it contains the desired object, i.e., performs the classification task, dividing the input into two classes. The landmark detection algorithm proposed by Dlib is an implementation of the Regression Tree Ensemble (ERT), introduced by Kazemi and Sullivan [10]. This method uses a simple and quick function to directly estimate the location of a landmark. These estimated positions are subsequently refined using an iterative process performed by a cascade of regressors. Regressors make a new estimate from the previous one, trying to reduce the error of alignment of the estimated points at each iteration. In our work, we decided to use the dlib library to determine the outline of the eye. At the first stage, the dlib.get frontal face detector() function determines the face contour. Next, using the dlib.shape predictor command ("shape predictor 68 face landmarks.dat") we define facial features. For finding the eye shape, we used the model trained for 68 landmarks. We only use points 36–41 and add 20 millimeters to each point to expand the range, and then use an OpenCV ROI to limit the area with the eye in the video stream. See Fig. 3, which shows images from video with the eye limited by a rectangle with resolution 416×416.



a b c

Fig.3. Detect eye in video stream. a – face detection, b – new video stream with points 36 to 41, c – expand video stream with points 36 to 41

4. Deep Learning for Gaze Detection and Datasets

Although there are a large number of different neural networks, for gaze tracking tasks where speed and high accuracy are required, only a few neural networks are suitable. Here, we considered YOLOv3, SSD, Mask RCNN, Freeznet. The main problem of deep neural networks is the need to use a large number of images to train the network. Several datasets can be used for non-commercial use to train neural networks:

- Columbia Gaze Data Set: This data set consists of 5,880 images of 56 people over varying gaze directions and head poses. For each subject, there are 5 head poses and 21 gaze directions per head pose (http://www.cs.columbia.edu/CAVE/databases/columbiagaze/);
- Rajeev et al. [22] used this dataset to train neural networks - Openeds facebook dataset;
- Aayushy et al. [1] used a semantic segmentation data set collected with 152 participants of 12,759 images with annotations at a resolution of 400×640. Although the challenge participation deadline was September 15, 2019, the dataset is still available upon request (https://research.fb.com/programs/openeds-challenge/);

- MPIIGaze dataset: This data set consists of images taken in everyday conditions using the laptop's built-in webcams, in which 15 people participate. The MPIIGaze dataset contains 213,659 images [27];
- Kaggle dataset, competition: Competitions are held periodically, participation in which open access to the dataset (https://www.kaggle.com/c/gl-eye-tracking);
- Image.net: A massive database of annotated images designed to test and test methods of pattern recognition and machine vision.

Unfortunately, we cannot use the known datasets, because our approach develops low-cost eye-tracking with deep learning functionality. In the described datasets the images at very high resolution can only be used for medical purposes. In our work, we used a custom dataset of 3000 images from 3 people. From previously described models for eyetracking we selected YOLOv3. This model on well-known datasets (Cifar, imagenet) showed good performance and tracking accuracy. In YOLOv3 objects are defined by a rectangular frame. Image marking was carried out using the YOLO-Annotation-Tool-master program (open source), an example of marking on the image is shown in Fig. 4.

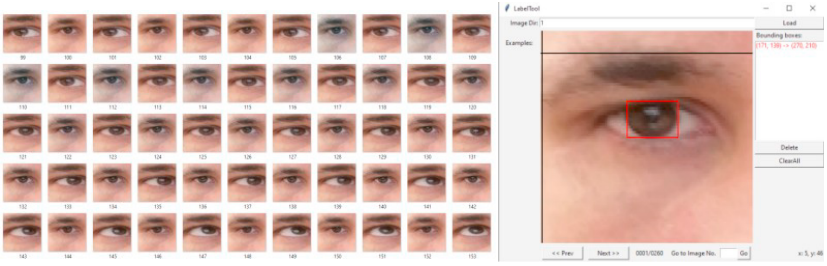


Fig. 4. Labeling with YOLO-Annotation-Tool-master program

In this research, we used the non-standard method of tracking gaze. Three models of the object for the only one gaze at once were used (Fig. 5). The computer accepts the coordinates of the object that has the highest probability of correct recognition. This method shows a more vivid effect when working with small data sets.

a b c d

Fig. 5. Labeling image for training model: a - the eye on the left, b - the eye in the middle, c - the eye on the, d - the inner and outer corners of the palpebral fissure.

The entire dataset from 3 people: 3000 images, 1000 images; the eye on the left, 1000 images; the eye on the right, 1000 images; the eye in the middle, 3000 images for the inner and outer corners of the palpebral fissure. The inner and outer corners of the palpebral fissure was used as a point of origin. Ultimately, in the quality of the points for the origin, only the inner palpebral fissure was chosen, since its tracking accuracy is higher, in Fig. 6, point 1.

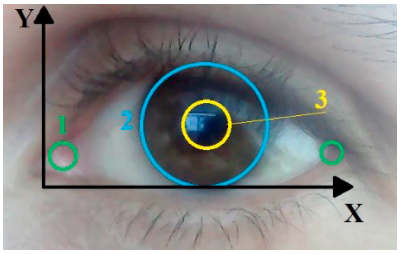
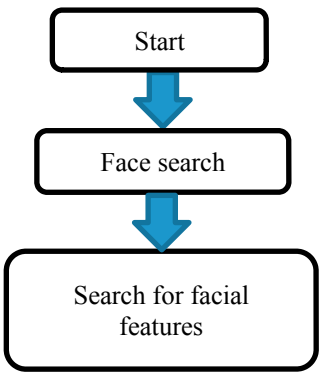


Fig. 6. The calculation of gaze position. Inner palpebral fissure, used as the origin point x, y: (0, 0): 1 - inner palpebral fissure, 2 – iris, 3 – pupil

YOLOv3 does not predict the absolute coordinates of the center of the bounding box. Rather, it predicts displacements, relative to the upper left corner of the grid cell that predicts the object. If the forecast for the center is (1, 1, 2, 3), then this means that the center is at the point (7, 1, 9, 3) on the 13×13 characteristics map. The bounding box dimensions are predicted by applying the log output space transform to the output, and then multiplication with reference. The resulting forecasts, bounding box width, and bounding box length are normalized in height and width of the image. Thus, if the length and width of the forecast for the field containing the eye are (0.1, 0.2), then the actual width and height on the characteristics map is (13×0.1, 13×0.2). An object's assessment is the probability that the object is contained within a bounding box. Class trusts represent the probabilities of a discovered object belonging to a class. Before version 3, YOLO used softmax to evaluate the class, but in the YOLOv3 - softmax YOLOv3 predicts 3 different scales. The detection layer is used to detect characteristics of three different sizes on the cards with steps 32, 16, 8, respectively. This means that when entering 416×416, we do detection at 13×13, 26×26 and 52×52 scales. The network reduces the sampling frequency of the input image to the first detection layer, where detection is performed using object maps of the layer in increments of 32. Also, the layers are sampled by a factor of 2 and combined with object maps of previous layers that have the same size object map. Now another detection is performed on the layer from step 16. The same upsampling procedure is repeated, and the final detection is performed on the layer of step 8. At each scale, each cell predicts 3 bounding boxes using 3 anchors, with the result that the total number of anchors used is 9. In this work, the anchors using k-means were calculated. As initial weights, we used 237 MB weights presented by the developers of the YoloV3 library. In the model, we froze the last 3 output layers and trained the last layers with our dataset. The authors of YoloV3 implemented a neural network on the Darknet framework, but we did it on Keras. In the final, we obtained 256 MB of weight. To test the implemented neural network model, we used a real-time situation. We achieved this result using a relatively small dataset, which is a good result for eye-tracking tasks.

5. Empirical results

In this work, the algorithm for eye-tracking was used as shown in Fig.7.



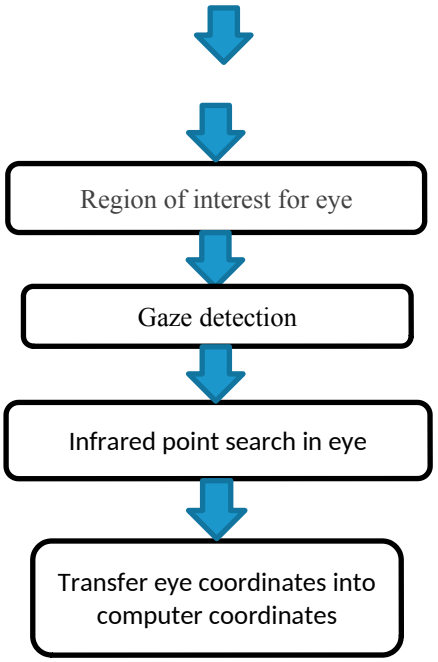


Fig. 7. The algorithm for eye-tracking

We developed the eye-tracking algorithm with two coordinate systems. The first coordinate system, due to OpenCV and the inrange function, determines the position of the eye relative to the infrared LED which is in front of the monitor. This system coordinates to control the position of a head relative to the monitor. The second coordinate system calculates the position of the eye due to YOLOv3 relative to the inner and outer corners of the palpebral fissure. The second coordinate system of the coordinates translates its coordinates into the first, thereby controlling gaze even when the head changes its position. Fig. 8 shows schematically the process of transferring gaze coordinates at the computer.

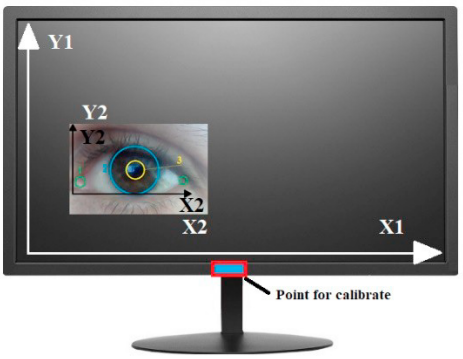


Fig. 8. Schematic of the process of transferring gaze coordinates at a computer

Fig. 9 shows the experimental device for eye tracking. Using a servomotor to control the position of the camera allows us to significantly increase the area and autofocus of the camera work range from 600×800 mm to 1000×1200 mm. The servo motor is controlled by a Raspberry PI3, which receives the coordinates of the eye using the TCP-IP protocol from the computer. The focus of the camera is controlled by a simple neural network for classification.

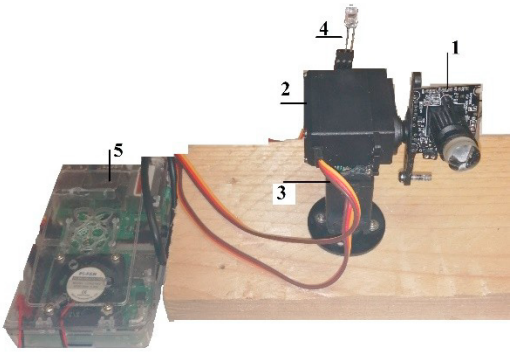


Fig. 9. Experimental setup: 1 - camera with a resolution of 1080, 5Mp (2560×1920), 2 - servomotor

We have 4 stages of classification quality (see Fig. 10).

1 2 3 4

Fig. 10. Examples of images used for a neural network to classify focus - from 1: good - to 4: bad focus

Raspberry PI3 receives classification result from the computer by TCP/IP every 0.5 sec and, depending on the result, controls the focus position. For the classification of quality of the image focus, a convolutional neural network (CNN) was created. The CNN consists of 8 layers of Convolution2D and MaxPooling2D layers after the second and fourth convolution. On all layers, the ReLU activation function is used. To regularize our model, after each subsample layer and the first fully connected layer, the Dropout layer was used. The neural network has four outputs that characterize the quality of focusing. In our approach, the YOLOv3 network is subsequently used, which on our dataset of 1,500 images showed a great effect in the process of tracking one object in comparison with other models. The device was tested for the condition when the Haar cascades and the dlib library determined the position of the face and eyes with 100% accuracies. The camera focus has been adjusted visually. Correspondingly, the entire error that was present during the tracking process is the error of the implemented neural network for tracking the eyes. Calibration of the device was carried out on 20 bright color squares, an area of 7.8 sm^2, located on the monitor screen in the form of a desktop screen saver. The subject concentrated attention for 3 seconds alternately on bright colored squares on the computer screen (see Fig. 11), the coordinates of each square were recorded in the computer's memory

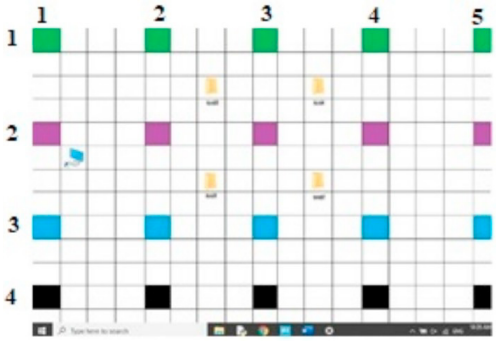


Fig. 11. Computer desktop screen saver to calibrate and verify the eye-tracking device.

Verification of the developed tracking device was carried out in comparing the coordinates obtained during the

tracking process with the coordinates that were obtained during the calibration process. The monitor size is 22 inches with a screen resolution of 1366×768 pixels. The subject was located 500 mm from the monitor in a natural position for work. The error of concentration of the subject's gaze on the monitor by one square gives an error of 2 degrees. The results are presented in Table 1. For each point, the average error for 20 measurements is given. The total error is 2.25 degrees. Frequency of the developed device - 30 Hz.

Table 1. Error in degrees for various locations on a computer monitor

Error in degrees					
Position	1	2	3	4	5
1	2.05	2.6	3.2	2.95	1.95
2	3.11	1.68	2.2	1.9	2.35
3	0.85	3.3	2.8	2.1	1.75
4	0.95	2.85	2.2	3.15	1.05

The training of the model took place in different lighting conditions, with and without light. Moreover, if a new user will work under different lighting conditions from the conditions presented here, it is only necessary to include images from this state in the dataset. In this work, we used a method where, due to a combination of various machine vision mechanisms and a nonstandard approach to marking, we managed to develop a budget device in the price category of no more than 30 dollars, table 2.

Table 2. Error in degrees for various locations on a computer monitor

YOLOv3 characteristic					
Name	Frame size, mm	Accuracy in degrees	Frecuency,hz	Weight, g	Cost, dollars
YOLOv3	1000×1200	2	30	300	25

Different lighting, reflections on the eye, camera resolution, and camera position can affect this accuracy of the developed device. To avoid these disadvantages, in the next research, we want to prepare new labeling images with different eye images (with a lot of subjects). As a result, the neural network will be trained to find the eye not so much in the color characteristics of the pupil, but in the color contrast between pupil and sclera. External factors have a passive effect on this feature

6. Conclusion and Discussion

Despite the more than 100-year history of studying the movement behind gaze, one of the main drawbacks of research in this area is the lack of any standards in the development of these devices. Part of the study considers the criterion of the accuracy of loss in degrees as a success criterion, where the accuracy of 1-2 degrees is considered nearly optimal. Other works describe calibration strategies, the invariable posture of the head and methods for assessing gaze. In future research, it is necessary to focus on standardization of research in the field of eye movement, which will allow more competent comparison of various research and give a more complete picture of the situation in the field of development of these technologies and thereby more intelligently present to researchers their developments. Moreover, despite the explosive growth of interest in the use of neural networks in image processing, in the described studies, as a rule, the authors use a standard convolutional network for deep learning or use the OpenCV library tools. The disadvantage of this tracking method is the need for labeling. At the same time, the YOLOv3 network is constructed in such a way that labeling is made only by a rectangle. For comparison, we obtained a higher tracking accuracy when using Mask-RCNN. The main disadvantage of the Mask-RCNN is the lack of speed, as a result of which it cannot work on the streaming video. The further task is to train the neural network from scratch without external weights trained on various images. To train the network need uses only for the eyes image in the extension 416 to 416 and leave only the last layers without freezing. This will allow a user to

ultimately use a small number of images for completely train the network.

References

[1] Aayushy, C., Rakshit, K., 2019. Ritnet: Real-time semantic segmentation of the eye for gaze tracking, 1–7
[2] Bueno, A., Sato, J., Hornberger, M., 2018. Eye tracking – the overlooked method to measure cognition in neurodegeneration? Neuropsychologia, 133, 107–109. doi:10.1016/j.imavis.2018.05.004
[3] Davin, P., Heather, L., 2013. How to build a low-cost eye-tracking system. Ergonomics in Design The Quarterly of Human Factors Applications, 20, 1300–1305. doi:10.1177/1064804611428928
[4] Frank, H., Carlos, H., Kim, J., Whang, M., 2019. Towards a low cost and high speed mobile eye tracker. Proceedings of the 11th ACM Symposium on Eye Tracking Research Application, 16, 1–9. doi:10.1145/3314111.3319841
[5] Hari, S., 2012. Human eye tracking and related issues: A review. International Journal of Scientific and Research Publications, 2, 1–9
[6] Huang, B., Chen, R., Zhou, Q., 2020. Applying eye tracking in information security. Pattern Recognition, 98, 145–157. doi:10.1016/j.patcog.2019.107076.
[7] Huang, H., Xu, Y., Hua, X., Yan, W., 2019. A crowdsourced system for robust eye tracking. Journal of Visual Communication and Image Representation, 60, 28–32. doi:10.1016/j.jvcir.2019.01.007
[8] Jagtap, N., Kolap, A., Adgokar, M., 2015. Real time driver's eye tracking design proposal for detection of fatigue drowsine. International Journal of Innovative Research in Computerand Communication Engineering, 3, 3758–3758. doi:10.15680/ijircce.2015.0305003
[9] Javier, S., Skovsgaard, H., 2010. Evaluation of a low-cost open-source gaze tracker, 22–24doi:10.1145/1743666.1743685
[10] Kazemi, V., Sullivan, J., 2014. One Millisecond Face Alignment with an Ensemble of Regression Trees, in: Proceedings of the 2014 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1867–1874
[11] Kerr, R., Marwan, M., Fuad, M., 2019. A real-time lazy eye correction method for low cost webcams. Procedia Computer Science, 159, 281–290. doi:10.1016/j.procs.2019.09.183
[12] Krafka, K., Khosla, A., Bhandarkar, S., 2016. Eye tracking for everyone. Journal of Neuroscience Methods, 274, 13–26.
[13] Larrazabal, A., Cena, G., Mart´ınez, C., 2019. Video-oculography eye tracking towards clinical applications: A review. computers in biology ´ and medicine. Computers in Biology and Medicine, 108, 57–66. doi:http://dx.doi.org/10.1016/j.patcog.2014.01.005.
[14] Larsson, L., Schwaller, A., Nystrom, M., Stridh, M., 2016. Head movement compensation and multi-modal event detection in eye-tracking ¨ data for unconstrained head movements. Journal of Neuroscience Methods, 274, 13–26. doi:10.1016/j.jneumeth.2016.09.005.
[15] Lee, J., Cho, C., Shin, K., Lee, E., 2012. 3d gaze tracking method using purkinje images on eye optical model and pupil. Optics and Lasers in Engineering, 50, 736–751. doi:0.1016/j.optlaseng.2011.12.001.
[16] Li, J., Li, H., Ume, W., Wang, H., 2020. Identification and classification of construction equipment operators' mental fatigue using wearable eye-tracking technology. Automation in Construction, 109, 103–113. doi:10.1016/j.autcon.2019.103000.
[17] and Liangchan, P., Lei, F., Xieping, G., 2018. Salient object detection based on eye tracking data. Signal Processing, 144, 392–397. doi:10.1016/j.sigpro.2017.10.019.
[18] Maurage, P., Masson, N., Bollen, Z., Hondt, F., 2020. Eye tracking correlates of acute alcohol consumption: A systematic and critical review. ´ Neuroscience Biobehavioral Reviews, 108, 400–422. doi:https://doi.org/10.1016/j.neubiorev.2019.10.001.
[19] Molina, A., Redondo, M., Lacave, C., Ortega, M., 2014. Assessing the effectiveness of new devices for accessing learning materials: An empirical analysis based on eye tracking and learner subjective perception. Computers in Human Behavior, 31, 475–490.
[20] Ozcelik, E., Karakus, T., Kursun, E., Cagiltay, K., 2009. An eye-tracking study of how color coding affects multimedia learning. Computers Education, 53, 445–453. doi:10.1016/j.compedu.2009.03.002.
[21] Pavlas, D., Lum, H., Salas, E., 2012. Low to build a low-cost eye-tracking system. ergonomics in design the quarterly of human factors applications. Ergonomics in design, 3, 18–21. doi:10.1177/1064804611428928.
[22] Rajeev, R., Shalini, D., 2018. Light-weight head pose invariant gaze tracking, 1–9.
[23] Robertson, E., Gallant, J., 2019. Eye tracking reveals subtle spoken sentence comprehension problems in children with dyslexia. Applied Psycholinguistics, 228, 102–105. doi:10.1017/S0142716409990208.
[24] Sun, J., Hsu, K., 2019. A smart eye-tracking feedback scaffolding approach to improving students' learning self-efficacy and performance in a c programming course. Computers in Human Behavior, 95, 66–72.
[25] Swarts, M., Noh, J., 2013. Ultra low cost eye gaze tracking for virtual environments. VAMR 2013. Lecture Notes in Computer Science 21, 1300–1305. doi:10.1007/978-3-642-39405-8_12.
[26] Viola, P., Jones, M., 2001. Rapid Object Detection using a Boosted Cascade of Simple Features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. I–I. doi:10.1109/CVPR.2001.990517.
[27] Xucong, Z., Yusuke, S., Mario, F., 2015. Appearance-based gaze estimation in the wild. Computer Vision and Pattern Recognition, 23, 456–459. doi:10.1016/j.procs.2016.07.013.