Pupil Segmentation and Tracking in Mouse Behavioral Study

Ha Yun Anna Yoon
Department of Mechanical Engineering
University of California, Berkeley
anna_yoon@berkeley.edu

Jaewan Hong
Department of Computer Science
University of California, Berkeley
jaewan@berkeley.edu

Abstract

Pupil tracking is time efficient and non-invasive method to study behavioral state compared to other *in-vivo* recording, fMRI, and two photon calcium imaging. To investigate the progress of neurological disease as well as visual and neurological functions, pupil tracking is a must. However, existing pupil tracking methods using CNN techniques require too much computational resources and human labeled data. This cost and labor-intensiveness impedes biologists in adopting CNN-based pupil tracking methodologies into their research. Thus we looked at utilizing traditional simple machine learning techniques to accomplish the job in more simply and quickly. Our method enables researchers to study and quantify gaze patterns and eye movements in an objective way that increases reliability and reduces variability. Our *Pupil Tracker* provides an ease-of-use and time-efficient method compared to the existing learning based tracking method like detectron2. Detectron2 takes 4 days to train with 4.5 s/frame for inference with 8 GPUs, and DeepLabCut takes 10 days to train with 0.094 s/frame for inference using a CPU. The Pupil Tracker, our in-house developed method using K-nearest neighbor, only takes 0.075 s/frame on a personal computer without any GPUs.

1 Introduction

In the field of neuroscience and psychology, pupil is often used to track behavioral state [1, 2]. Pupil tracking is critical because it enables scientists to determine whether animal is alert and attentive, sleeping, or in pain (squinting or crying), and etc [3]. With pupil area determination, scientists can also correlate certain neuronal response to pupil constriction and dilation that denotes the state of the animal[4]. It is also critical in humans to track awareness when patients are in vegetative state post-accidents[5, 6]. In recent years, pupil tracking has also grown in importance for AR and VR as well as for biometrics mechanisms, such as ones used for unlocking electronic devices [7, 8, 9].

In the past, many algorithms focused on using image histogram values to determine the periphery of the pupil. With recent boom of neural networks, neural nets are widely used in any applicable areas. Recent developments in CNN has led to more methods geared towards deep learning. There has been growing number of CNN-based image segmentation tools developed in the recent decade. For image segmentation and object tracking, Facebook AI research has developed a detectron2, which is based on Mask R-CNN [10, 11]. For 2D and 3D marker-less pose estimation, Mathis Lab has developed DeepLabCut, which is based on transfer learning on deep neural network [12].

In neuroscience, scientists acquire pupil data in various different conditions. Since each lab and experiment has its own sets of environmental conditions, it is time-consuming and labor intensive to use any deep learning methods. We have tested the conventional image segmentation tools available in the status quo, such as detectron2 and DeepLabCut. Based on our experience we have found that CNN-based approaches require people to train for each set of conditions with thousands of labeled

data, which impedes the processing of the data. Thus, we developed Pupil Tracker, which uses simple K-nearest neighbor algorithm to segment and track pupil without the need to train a neural network. By scoping down the focus area to search the pupil, an eye image can be binarized by its nature. There exists only a few color schemes near the eye which can be binarized to make the problem simple. Then the binarized image can easily be classified using K-nearest-neighbor algorithm [13]. We implemented Pupil Tracker from scratch with 3500 LoC using Matlab [14] for ease of use by scientists. We made it easy-to-use with easily adjustable parameters for scientists with limited coding background to process all acquired pupil data. No labeled data and expensive accelerators (GPUs and ASICs) are required. Pupil Tracker requires a single manual input per video and can be run on minimal computing power.

2 Challenges in Pupil Detection

Despite vast need for consistent pupil tracking in these fields,current algorithm to segment pupil from acquired images are far from fail proof. There are many challenges when detecting the pupil accurately. Figure 1 shows examples of pupil segmentation failure by the conventional methods. If the subject's *pupil dilates* over the capacity of tractable cutoff, machine fails to accurately detect the pupil outline. *Dynamic movement of subject*, such as head jerking motions, places eyes out of machine's observatory scope. Patches of *light reflection* on the pupil, especially when we are looking at visual experiments, blurs the boundary of eyes, hindering machines from accurately defining the pupil boundaries. Conventional machine learning tactics fail when other features, such as *whiskers* and *mouse paws*, get in the way. Similarly, squinting or sleeping animal data also result in failure of accurate detection by the machine. Animals often *blink eyes* which add noise to sample data. Each lab + experiment uses different brightness screen for visual stimulus or for eye-recording camera parameters (exposure etc.), so it is impossible to use a single trained model for all the different experiment types. Thus, it is critical that we stay away from methods that require a huge amount of training time.

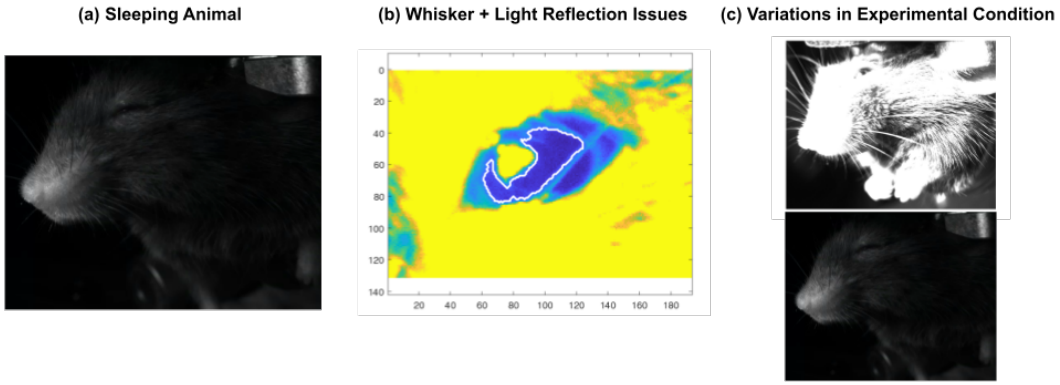


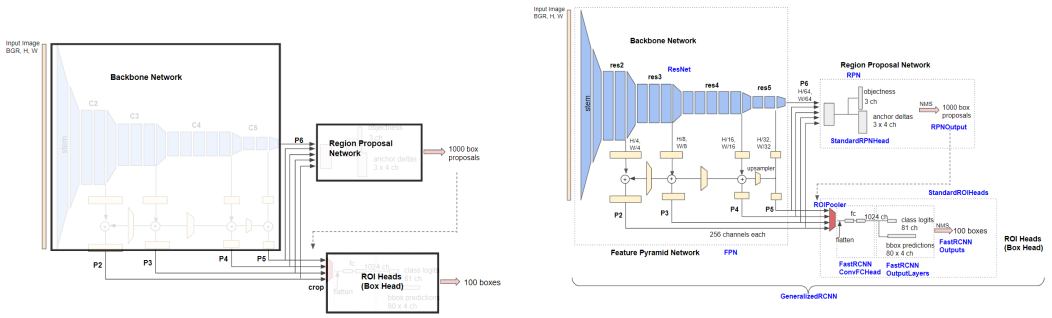
Figure 1: Issues with Current Pupil Segmentation and Tracking Methods

3 Related Works

3.1 Detectron2

Detectron2 is an open-source object detection system developed by the Facebook (now, Meta) AI research team. Detectron2 is widely used in object detection and segmentation tasks. It is based on Fast Region-Based Convolutional Neural Networks (R-CNNs) [15]. Fast R-CNN is trained as a single model instead of three separate modules in R-CNNs. In Detectron2, using Fast R-CNN improves the training and predicting time, but the region proposal inputs have to be done separately.

Figure 3 explains the architecture of Detectron2. It consists of three stages: backbone network, region proposal network, and Box head. Detectron2 architecture takes the images and proposes ROIs. It then passes them through pretrained ResNet or VGG-16 models to extract features [16**?**]. The backbone



(a) Meta Architecture of Base-RCNN-FPN (from [17])

(b) Detailed Architecture of Base-RCNN-FPN (from [17])

Figure 2: Detectron2 Architecture

network extracts features from the input image and outputs features. Extracted features are passed to region proposal network consisting of ROI pooling layer, two fully connected layers, and other two fully connected heads for regression and label classification. This network outputs box proposals with confidence scores. For evaluation purposes, detectron2 uses mean average precision.

Figure 3: Image detection using Detectron2

3.2 DeepLabCut

The Mathis Lab at EPFL developed DeepLabCut in 2018 [12]. It was designed for 2D and 3D markerless pose estimator based on transfer learning with deep neural networks. It is now widely used in biological applications to study the motion of fruit flies, mice, and zebrafish etc.

DeepLabCut is designed for object recognition and semantic segmentation. It uses pretrained ResNets and deconvolutional layers. In our particular model, we initialized with a trained ResNet101, which has 48 convolution layers and 1 of each MaxPool and AveragePool layers. Resnet weights were trained on ImageNet, a large-scale object recognition benchmark [**?**]. After ResNets, deconvolutional layers are added to up-sample the visual information. For each ROI, its spatial probability density represents that it is in a particular location. In each iteration, the network assigns high probabilities to labeled ROI locations and low probabilities elsewhere.

To make our model, we first choose 200 frames pulled from ten data sets. Then, as Figure 4a shows, we choose ROI to be the eye region. We then hand-labelled with four markers the left, right, top, and bottom of the pupil on all of the 200 frames as shown in Figure 4b. The labelled data is trained on a deep neural network consisting of ResNets and deconvolutional layers as explained above and

in Figure 4c. Once it trains for a few days–approximately 150,000 iterations on a CPU, the trained model can then be used to predict labels on other mice pupil videos as seen in Figure 4d.

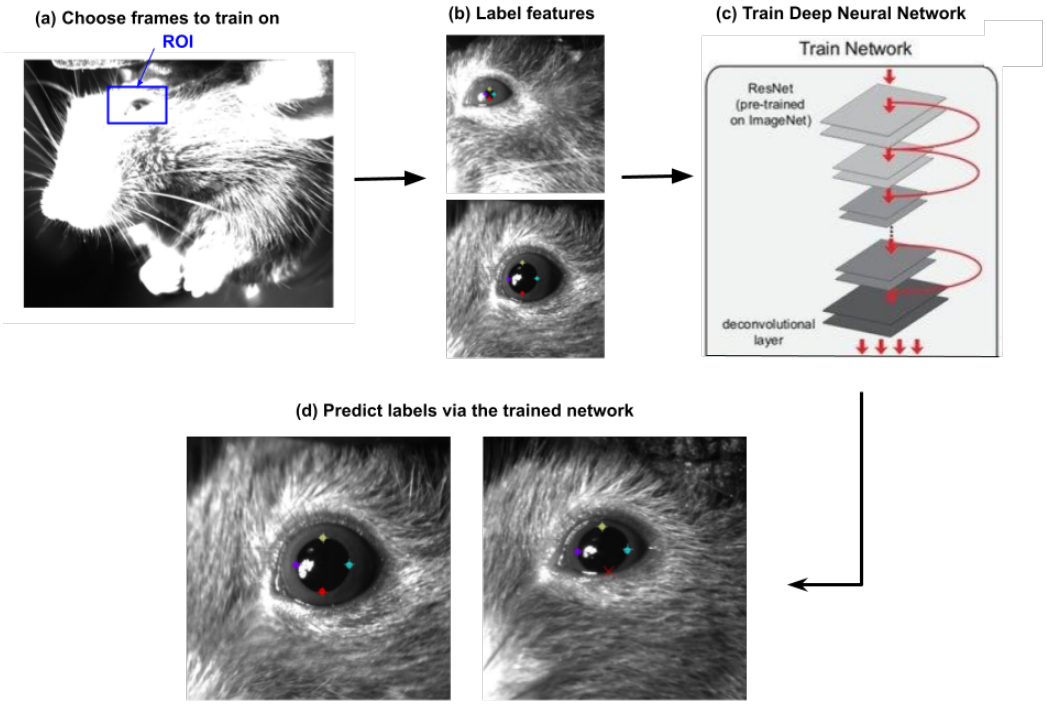


Figure 4: How DeepLabCut works

4 Pupil Tracker

Figure 5 illustrates our pipeline. Pupil Tracker is designed to operate with the minimal computational resources. Since most neuroscientists are comfortable with MATLAB and have only CPU available, we fit our design to use these resources.

```
// Manually labeled window coordinates at first frame of the video to start from
image_coordinates = Get_Image_Window();
simple_grey_and_black_images = Crop_To_Window(video_file, image_coordinates);
binarized_frames = Scale_and_Binarize_Image(simple_grey_and_black_images);
for(frame: binarized_frames){
    center_of_eye, radius = K_Nearest_Neighbor_Image_Binary(frame);
    //Assume the pupil is circle
    pupil = Circle_Pupil(center_of_eye, radius);
    while(NotCircular(pupil)){
        pupil = Find_Elliptical_Mask(frame, center_of_eye, radius);
    }
}
```

Listing 1: Pupil Tracker Algorithm

Our algorithm for Pupil Tracker is outlined in Listing 1. First, we input the animal face video data. The full video is then segmented to get a window of only the eye area as seen in Figure 5a. We scale the window pixel values and binarize the image to get a threshold to differentiate between the pupil

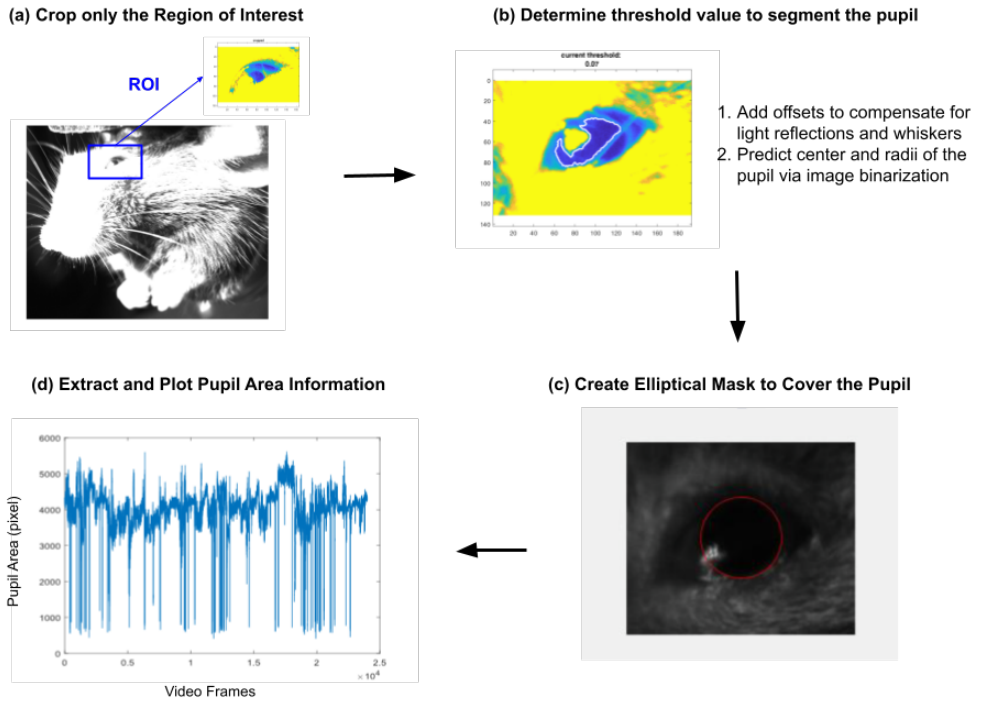


Figure 5: Visualization of the data sets. (a)A frame from animal face video. (b) Cropped eye image from the animal face video. (c) An example of accurate pupil tracking and segmentation. (d) Final output from the algorithm that enables pupil area plots to visualize the pupil constriction and dilation changes within the duration of the video.

and non-pupil area. When the threshold is done correctly, it accurately segments the periphery of the pupil from rest of the eye. As seen in Figure 5b, the threshold often fails when there is a path of light reflection or whisker in the way. Thus, when we have the threshold value to accurately segment the periphery of the pupil, we provide the visualization to the user to determine whether the value is acceptable. In cases where we do have impediments, offsets are added to compensate for the reflections and whiskers and then use our resulting k-nearest neighbor algorithm to find the center and radius of the pupil. When the knn results are circular, we simply create a circular mask based on the data. However, when the previous result is not circular based on our circularity test, our algorithm looks for and creates an elliptical mask by testing a range of radius and offset values. As a result, our algorithm enables tracking and segmentation of the pupil through the entire duration of the video like in Figure 5c. Our final output is a video and functional values that can be used to create plots similar to Figure 5d that visualizes the functional data, such as pupil area, center, and the radii of the elliptical mask. Extract functional information is very useful for behavioral analysis, especially when looking at correlation with the neuron activity.

5 Results

5.1 Detectron2

Amount the three methods, detectron2 required the most exhaustive computational resources. First, we ran the inference demo on their pre-trained models to find out the time it takes to run each inference tasks. For their pre-trained model, inference took 4.5 seconds on average, and their model was trained for 4 days using 8 NVIDIA V100 GPUs and NVLink.

Second, we tried to train a model for mice pupil tracking using a cpu. Many neuroscientists would only have cpu on their work stations, and even if they have a GPU, it would not be the good quality GPU that is needed for quicker training setup. Thus, we set cfg.MODEL.DEVICE='cpu' to run detectron2 with CPUs only. However, we prematurely stopped training after a day on our personal computer with CPUs only. Based on Facebook AI Labs, it would take about 4 days with 8 GPUs to train 200 label frames for 100,000 epochs. To train for accurate detection, it would be time and labor-intensive. The biggest disadvantage with detectron2 is that it requires labeled features on images that are not pre-trained and distributed by Meta. In our case, we had to label hundreds of mice images with pupil labeled. In addition to manually labeling frames, they also have to train the network for the next couple of days, which hinders their usage of the computer during that time. We concluded that this loss of time and computer usage would not be reasonable for many neuroscientists in academia.

5.2 DeepLabCut

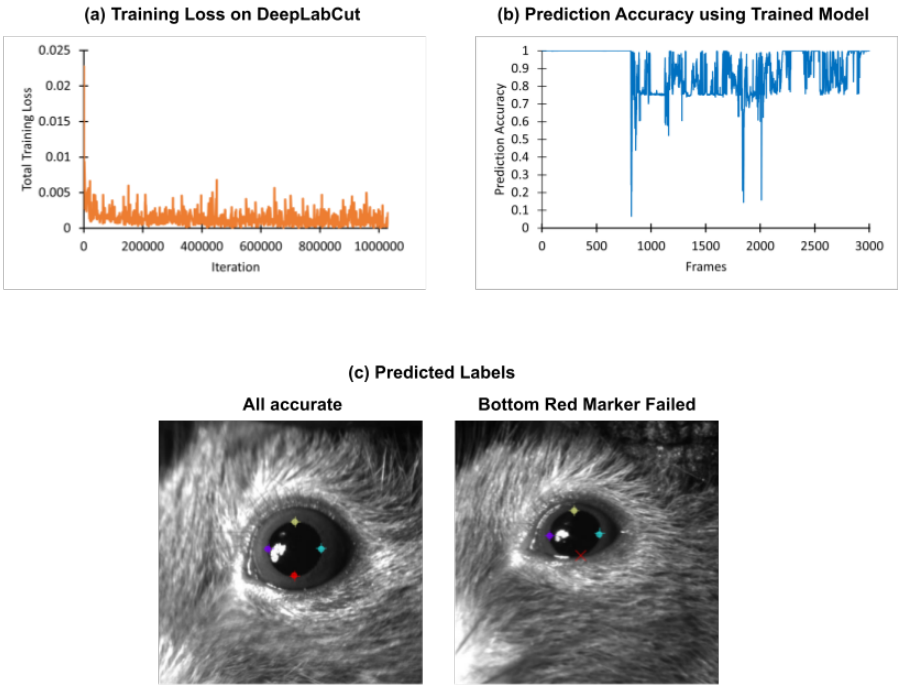


Figure 6: Training and Prediction using DeepLabCut. (a) Training loss on DeepLabCut. We trained 200 labelled frames for 1,030,000 iteration that took 10 days on CPU. The training loss converged to 0.0001 with training error of 0.99 and test error of 1.32. (b) Prediction accuracy using the trained model. Overall, the prediction accuracy is approximately between 0.8 and 0.9. (c) Predicted labels on few example frames. The example on the left shows all accurate features while the one on the right shows the red marker on the bottom failing to detect the boundary correctly.

Using DeepLabCut, we trained 200 labelled frames, pulled from ten experimental data videos, for 1,030,000 iterations. The entire training time on a CPU took ten days. Figure 6 shows the training loss and prediction accuracy logs of the model. The final training loss was 0.0001 with training error and test error being 0.99 and 1.32, respectively. For the last few days, the loss rate did not improve. Thus, we conclude that approximately four days of training with 150,000 iterations is enough to get a similar quality model. On this trained model, inference took 0.094 s/frame. Overall, the prediction accuracy is approximately between 0.8 and 0.9, which is not the best prediction. From predicted labels, we can see that there are occasionally one or two failed labels when the pupil is covered by the eye boundaries. For example, when the animal is squinting or the pupil is dilating, we can see

Table 1: Results Comparison among the three methods: Detectron2, DeepLabCut, and Pupil Tracker.

	Detectron2	DeepLabCut	Pupil Tracker
Method	CNN(ResNet101)	DNN	K-Nearest Neighbor
Train Time	$\sim$4 days (100,000 epochs)	$\sim$4 days (150,000 iter)	-
Pred Time	4.5 s / frame	0.094 s / frame	0.075 s / frame

the eyelids covering part of the pupil, which makes the machine unable to detect the boundaries correctly. This boundary issue is consistent with other conventional ML-based methods in the status quo. However, DeepLabCut works well with light reflection challenges.

5.3 Pupil Tracker

In Pupil Tracker, we used MATLAB codes that is easy-to-use with adjustable parameters for all scientists to use with mere CPU. Our method does not need to train since it is not neural network based. With Pupil Tracker, we looked at four private data sets of mice face video that were acquired in mouse behavioral studies. To process 24000 frames, it runs for 30 minutes to accurately predict the pupil segmentation. Thus, the processing speed is 0.075 s/frame. Pupil Tracker is great because the 24000 frame video was experimentally acquired at 15 fps, so the actual preprocessed video length is approximately 27 minutes. For our method to take only 30 minutes to completely process everything correctly, it is comparable with the initial preprocessed video length time.

6 Discussion

In this project, we developed our ease-of-use Pupil Tracker that runs on CPU. We simplified the algorithm by using k-nearest neighbor instead of any deep learning methods. We also compare our Pupil Tracker with two other pre-existing and widely used CNN-based image classification packages: Detectron2 and DeepLabCut. Table 1 shows the comparison of the methods, training time, and prediction time for all the three methods discussed above. Detectron2 takes 4 days to train with 4.5 s/frame for inference with 8 GPUs. DeepLabCut takes 10 days to train with 0.094 s/frame for inference using a CPU. The Pupil Tracker only takes 0.075 s/frame on a personal computer with mere CPU. Therefore based on just time aspect, Pupil Tracker is the best since we do not have have to train. Our processing speed is also much faster (approximately 60 times) than Detectron2's inference time and slightly faster than DeepLabCut's inference time by 0.02 s/frame.

As of the previous issues with light reflections and whiskers in the way, Pupil Tracker tackles those challenges by adding offsets when creating elliptical masks for pupil. Both DeepLabCut and Detectron2 overcame the light patch reflection challenges by training the models. However, DeepLabCut still runs into issues when the pupil is dilated or covered by the eyelid. When the pupil dilates past the capacity of tractable cutoff, DeepLabCut still fails to accurately detect the pupil outline. The biggest downsides to the CNN-based packages, such as DeepLabCut and Detectron2, are that they require training on labelled data. Since lighting and environmental conditions change for each setup and experiment, it would mean that the scientists would need to hand-label and train the sets every time they change their setup. Training alone is very time-consuming and labelling would add to labor-intensive aspect. Thus, for scientists using CPUs to analyze their data, CNN-based methods is not very attractive in an academic level.

As stated before, pupil tracking is critical measure of behavioral state in neuroscience and psychology. There is a growing desire to apply pupil tracking technology in pharmaceuticals to understand the effects of certain drugs on brain activity. Especially for those who cannot communicate–i.e., patients in vegetative states, babies, or even people with speech impairments–pupil tracking technology could provide a great insight into the alertness of the brain in different situations.

In non-biological fields, pupil tracking is used for AR and VR technology as well as for electronics unlocking mechanisms via biometrics. This technology is also beneficial in understanding people's shopping habits in terms of what people are looking at the most during shopping. It can be also used for surveillance of ex-offenders to prevent crimes by tracking their pupil movements.

With the various applications of pupil tracking, we believe that our Pupil Tracker will provide a very simple solution to quick pupil tracking that could applied to many different environments. We hope that there will be great development of non-CNN based pupil segment and tracking methods to uncover the mysteries of neuron activity and people's behavioral states in the future.

Acknowledgement

Thanks to GSI Anastasios Angelopoulos for shepherding our project and informing us of the state-of-the-art eye tracking methodologies. The core algorithm of this work was implemented in collaboration with Yuhan Yang. Emma Fortmann helped us to compare Pupil Tracker with DeepLabCut.

References

[1] Alexandra Wolf and Kazuo Ueda. Contribution of eye-tracking to study cognitive impairments among clinical populations. *Frontiers in Psychology*, 12, 2021.

[2] Roy S. Hessels and Ignace T.C. Hooge. Eye tracking in developmental cognitive neuroscience – the good, the bad and the ugly. *Developmental Cognitive Neuroscience*, 40:100710, 2019.

[3] Tad T Brunyé, Trafton Drew, Donald L Weaver, and Joann G Elmore. A review of eye tracking for understanding and improving diagnostic interpretation. 4, 2019.

[4] K Krejtz, AT Duchowski, A Niedzielska, C Biele, and I Krejtz. Eye tracking cognitive load using pupil diameter and microsaccades with fixed gaze. *PLoS ONE*, 13, 2018.

[5] Alexandra Vassilieva, Markus Harboe Olsen, Costanza Peinkhofer, Gitte Moos Knudsen, and Daniel Kondziella. Automated pupillometry to detect command following in neurological patients: A proof-of-concept study. *PeerJ*, 7, 2019.

[6] Marion Quirins, Clémence Marois, Mélanie Valente, Magali Seassau, Nicolas Weiss, Imen El Karoui, Jean-Rémy Hochmann, and Lionel Naccache. Conscious processing of auditory regularities induces a pupil dilation. *Scientific Reports*, 8(1), 2018.

[7] Anastasios N. Angelopoulos, Julien N. P. Martel, Amit P. S. Kohli, Jörg Conradt, and Gordon Wetzstein. Event based, near eye gaze tracking beyond 10, 000hz. *CoRR*, abs/2004.03577, 2020.

[8] Jia Zheng Lim, James Mountstephens, and Jason Teo. Eye-tracking feature extraction for biometric machine learning. *Frontiers in Neurorobotics*, 15, 2022.

[9] Ahmad F. Klaib, Nawaf O. Alsrehin, Wasen Y. Melhem, Haneen O. Bashtawi, and Aws A. Magableh. Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and internet of things technologies. *Expert Systems with Applications*, 166:114037, 2021.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[11] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.

[12] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, 2018.

[13] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[14] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[15] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[17] Hiroto Honda. Digging into detectron 2, Jan 2022.