Pistol: PUpil INvisible SUpportive TOOl to Extract Pupil, Iris, Eye Opening, Eye Movements, Pupil and Iris Gaze Vector, and 2D as Well as 3D Gaze

Wolfgang Fuhl[1,*], Daniel Weber[1] and Shahram Eivazi[1,2]
[1]*University Tübingen, Sand 14, Tübingen, Germany*
[2]*FESTO, Ruiter Str. 82, Esslingen am Neckar, Germany*

Keywords: Eye Tracking, Pupil, Gaze, Eye Ball, Eye Opening, Iris.

Abstract: This paper describes a feature extraction and gaze estimation software, named *Pistol* that can be used with Pupil Invisible projects and other eye trackers (Dikablis, Emke GmbH, Look, Pupil, and many more). In offline mode, our software extracts multiple features from the eye including, the pupil and iris ellipse, eye aperture, pupil vector, iris vector, eye movement types from pupil and iris velocities, marker detection, marker distance, 2D gaze estimation for the pupil center, iris center, pupil vector, and iris vector using Levenberg Marquart fitting and neural networks. The gaze signal is computed in 2D for each eye and each feature separately and for both eyes in 3D also for each feature separately. We hope this software helps other researchers to extract state-of-the-art features for their research out of their recordings.

1 INTRODUCTION

Eye tracking has a variety of application areas as well as research areas in today's world. In human-machine interaction it is used as a new input signal (Gardony et al., 2020; Arslan et al., 2021), in computer graphics as a constraint for the area to be rendered (Walton et al., 2021; Meng et al., 2020), in medicine using more eye features as data for self-diagnosis systems (Joseph and Murugesh, 2020; Snell et al., 2020; Lev et al., 2020) or to measure the eyes (Nesaratnam et al., 2017; Economides et al., 2007), in the field of psychology it is used to detect neurological diseases such as Alzheimer's (Davis and Sikorskii, 2020; Pavisic et al., 2021), in behavioral research to evaluate expertise as well as train students (Panchuk et al., 2015; Jermann et al., 2010), and many more like driver monitoring in autonomous driving (Liu et al., 2019; Shinohara et al., 2017) etc. This wide range of research and application areas requires that new eye features as well as more robust algorithms be easily and quickly deployable by everyone. As eye tracking itself is the focus of research and new more robust or accurate algorithms are constantly being published as well as new features, properties or signals such as pupil dilation are added, it is necessary that everyone has quick access to it to improve their products or integrate it in their research.

This poses a problem for the industry, since it must be determined long before the final product which functionalities and features will be integrated into the software (Lepekhin et al., 2020). If new features are added, this is usually postponed until the next generation of eye trackers. This is due to the fact that the industry's software must be very reliable and everything should be tested multiple times, as well as automated test cases for the new parts of the software must be integrated (Taylor et al., 2020). Further testing must be done with other integrated components, and the software must continue to be compatible with external software components used or developed by industry partners (Jiang et al., 2020). Another important point for the industry is the software architecture as well as also the quality of the source code. New parts must adhere to the software architecture as well as also be written in a clean and understandable way. This also delays the integration of new components, since the source code must also be checked.

Research groups themselves do not have these problems, since the software architecture is usually created and extended dynamically, which however also leads to a poorer source code quality (Hasselbring et al., 2020). Likewise, research groups have

*Corresponding author

Fuhl, W., Weber, D. and Eivazi, S.
Pistol: PUpil INvisible SUpportive TOOl to Extract Pupil, Iris, Eye Opening, Eye Movements, Pupil and Iris Gaze Vector, and 2D as Well as 3D Gaze.
DOI: 10.5220/0011607200003417
In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 2: HUCAPP*, pages 27-38
ISBN: 978-989-758-634-7; ISSN: 2184-4321

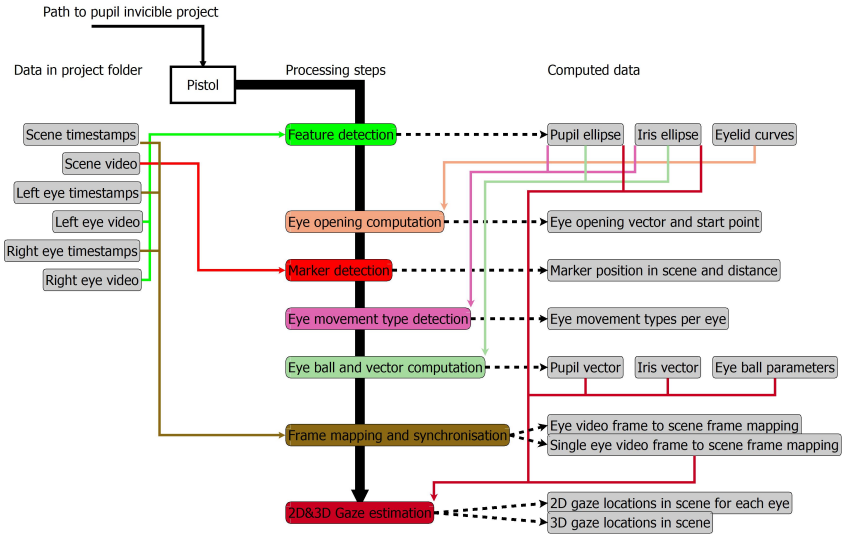

Figure 1: The workflow of Pistol. In gray are the data sources, and the single processing steps are in color. Each arrow corresponding to a data dependency is colored in the same color as the processing step.

the possibility to accomplish theses for the advancement of the software, whereby no costs develop. In the case that a research group develops such software, many new algorithms are already available in working form, which makes integration much easier and faster. Also, research groups do not have contractual partners, so they do not have to maintain the output formats and integrated interfaces in the software in a prescribed format for many years.

The Pistol tool presented in this paper extracts a variety of eye features such as pupil ellipse, iris ellipse, eyelids, eyeball, vision vectors, and eye-opening degree. These are needed in many applications such as fatigue or attention determination or can serve as new features in research, however these features are not provided by every eye tracker manufacturer. In addition, to the extracted features, our software offers the possibility to determine the eye gaze by different methods and based on different data like the iris or the pupil with the coordinates of the center or with the vectors. This also offers new possibilities in interaction and behavioral research. We hope that this tool will be useful for scientists all over the world and also help industry to integrate new features into their software. The Pistol tool will continue to be developed in the future, and we hope to be able to integrate many scene analysis techniques.

Our contributions to the state of the art are:

1. A free to use tool to extract a plethora of eye features for the pupil invisible offline.
2. 2D gaze estimation per eye with multiple optimization algorithms, as well as for the pupil and the iris separately.
3. 3D gaze estimation with both eyes using different optimization algorithms and separately for the pupil and the iris.
4. The eye trackers Dikablis, Emke GmbH, Look, Pupil, and many more are also supported for eye feature extraction and gaze estimation. But here the dirct path to the videos has to be specified.

2 RELATED WORK

In the field of eye tracking and feature extraction, there is a wide range of related work. This is due, on the one hand, to the industry, which provides a wide range of eye trackers at different prices and with different features. On the other hand, it is due to the ever-growing research community and the growing application fields for eye tracking.

From the industry there are for example the manufacturers Tobii (Tobii, 2021), ASL (ASL, 2021), EyeTech (EyeTech, 2021), SR Research (SRResearch, 2021), Eyecomtec (eyecomtec, 2021), Ergoneers (Ergoneers, 2021), Oculus (Oculus, 2021), Pupil Labs (PupilLabs, 2021), iMotions (iMotions, 2021), VIVE (VIVE, 2021) and many more. The eye trackers differ in frame rate and in the features used for eye tracking, each having its advantages and disadvantages. For a good overview of more details, please refer to the manufacturer pages as well as survey papers that deal with this overview (Rakhmatulin, 2020; Duchowski, 2002; Park and Yim, 2021; Mao et al., 2021).

Science itself, has of course produced some eye

trackers and systems for gaze calculation. The first one mentioned here is the EyeRec (Santini et al., 2017) software, which can be used for online eye tracking for worn systems. It has several built-in algorithms and uses a polynomial fitting to the optical vector of the pupil. Another software dealing with pupillometry is PupilEXT (Zandi et al., 2021). This is a highly accurate extraction of the pupil shape, which can also only be performed under severe limitations and with high-resolution cameras. The software OpenEyes (Li et al., 2006) offers a hardware and software solution in combination. The algorithm used for pupil detection is Starburst. For the Tobii Glasses there is also a tool (Niehorster et al., 2020), which allows analyzing the images offline and to apply algorithms from science to the data. For Matlab there is also a freely available interface to use Tobii Eye Tracker directly (Jones, 2018). For Tobii there also exists a custom software to improve the accuracy of the eye tracker (Phan, 2011). To distribute the data of a Gazepoint Eye Tracker over a network, there is also a tool from science (Hale, 2019). For studies with slide shows, there is the software OGAMA (Voßkühler et al., 2008). This can be used to record mouse movements and gaze signal and then analyze them, as well as create visualizations. GazeCode (Benjamins et al., 2018) is a software that allows mapping eye movements to the stimulus image. The software was developed to speed up the processing time for mapping and to improve the usability compared to commercial software like Tobii Pro Lab.

The distinguishing features of our software from existing ones is that we output a variety of other features such as pupil, iris, eyelids, eye-opening, pupil vector, iris vector, eye movements and different methods for gaze estimation. Also, we determine the 3D gaze point position and support a variety of eye trackers. Another important feature of our tool is, that each feature is exported together with a video, which showcases the quality.

3 METHOD

Pistol is executed by calling the program with a path to the Pupil Invisible project. Then you specify the recording to be processed (psX or just X after the usage of the Pupil Player) and Pistol starts the detections and calculations. In addition, you can specify the range of marker detection to be used for calibration (based on the scene video frames). If you do not specify this range, all detected markers will be used. Pistol also supports to specify a polynomial (or exponential function) and neural newtork architectures for the gaze estimation. For the eye ball computation, the window size to compute the eye ball can be specified as well as two diffrent approaches can be usd (Direct on one ellipse or the eye ball computation over multiple ellipses). If pistol is not used together with Pupil Invicible eye tracker recordings, the path to the videos has to be selected (Eye and scene videos).

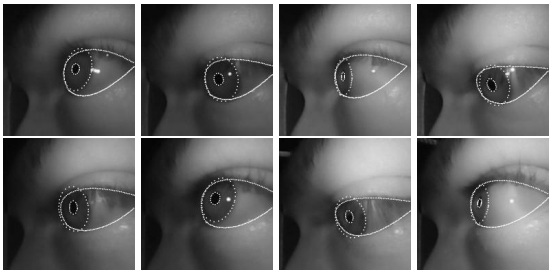

Figure 2: Exemplary detections of the pupil, iris, and eyelid for one subject. The first four images are from the left eye and the last four images are from the right eye.

Figure 1 shows the processing flow of Pistol and the data dependencies of each step. The gray boxes represent data that either exists in the project or was generated by a calculation step. Each calculation step in Figure 1 has a unique color, with which the data dependencies are also marked. At the end of each calculation step, the data is saved to a CSV file and a debugging video is generated to check the result.

In the following, we describe each processing step of our tool in detail. Some sections like the pupil, iris, and eyelid detection are combined since the fundamental algorithmics are similar.

3.1 Pupil, Iris, and Eyelid Detection

In Figure 2 we show some results of pupil, iris, and eyelid detection. For detection, we use small DNNs with maximum instead of residual connections (Fuhl, 2021a) as well as tensor normalization and full distribution training (Fuhl, 2021b) to detect landmarks. Using the maximum connections allows us to use smaller DNNs with the same accuracy. The full distribution training makes our DNNs more robust, and we also need less annotated data to train them. Tensor normalization increases accuracy and is more stable than batch normalization. In addition, we use landmark validation as well as batch balancing from (Fuhl and Kasneci, 2019) to evaluate the accuracy of the landmarks at the pixel level and discard detections if the inaccuracy is too high. To obtain the ellipses for the pupil and iris from the landmarks, we use the OpenCV (Bradski, 2000) ellipse fit. The shape of the upper and lower eyelid is approximated using cubic splines. The architecture of our models as well as the training details are in the supplementary material.

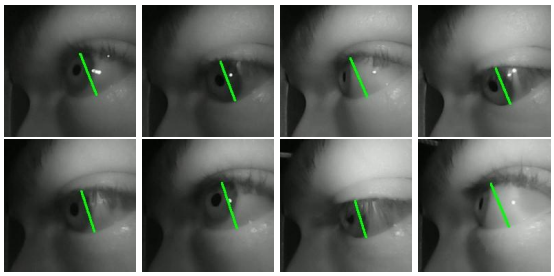

Figure 3: Exemplary images for the left (first 4) and right (last 4) eye of one subject. We compute the maximal distance along the vector between the eye corners to compensate for the off axial camera placement.

3.2 Eye Opening Estimation

Figure 3 shows some results of our eye-opening degree calculation. The eye-opening is calculated using an optimization procedure and is always oriented orthogonally to the vector between the eye corners. Without the constraint of this orthogonality (simple selection of the maximum over all minimum distances of the points on the eyelid curves), the results are very erratic, because due to the perspective of the camera, parts of the eyelids are not completely visible. Also, the orthogonality to the vector between the eye corners is what is expected in a frontal view of the eye.

Let P_{up} be the set of points of the upper eyelid, P_{dw} be the set of points of the lower eyelid, and $\overrightarrow{C}$ be the vector between the corners of the eye.

$$\text{Opening}(P_{\text{up}}, P_{\text{dw}}, \overrightarrow{C}) = \arg\max \arg\min |\overrightarrow{ud}|, \quad u \in P_{\text{up}},\ d \in P_{\text{dw}},\ \overrightarrow{ud} \perp \overrightarrow{C} \tag{1}$$

Equation 1 shows our optimization to compute the eyelid opening, where u and d are selected elements of P_{up} and P_{dw} with the side condition, that the vector between the two is orthogonal to the vector between the corners of the eye, i.e. $\overrightarrow{ud} \perp \overrightarrow{C}$. For a frontal image this additional side condition is usually not necessary, but for the images from the pupil invisible the results without this side condition are unstable and not always oriented correctly regarding the eye.

3.3 Eye Ball, Pupil&Iris Vector Estimation

To calculate the eyeball and the optical vectors, we use the neural networks of (Fuhl et al., 2020a). Here, several pupil ellipses are given into the neural network from which the eyeball radius and the eyeball center are calculated. As neural network, we used a network with one hidden layer consisting of 100 neurons. To calculate the optical vectors, we calculated the vector between the center and each of the pupil center and the iris center and converted it to a unit vector.

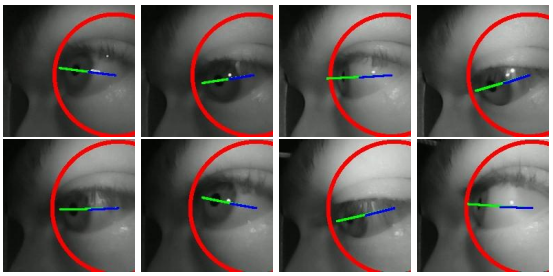

Figure 4: Exemplary images for the eyeball as well as iris and pupil vector. The eyeball is drawn in red, the iris vector in blue, and the pupil vector in green.

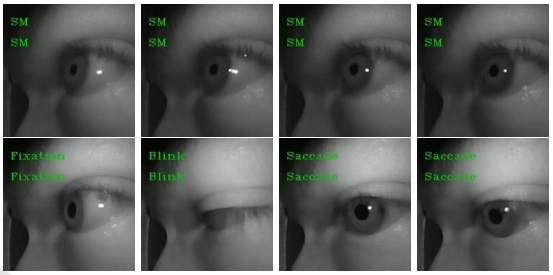

Figure 5: Exemplary detections for all eye movement types, Fixation, Saccade, Blink, and smooth pursuit. All images are from the left eye of the subject. In addition, if no feature is detected or only the eyelids are valid, but the eye is still open, the eye movement type will be marked as error. The four smooth pursuit (SM) images are from four consecutive frames with a distance of 25 frames (Eye cameras have 200FPS). The two saccade images are from two consecutive frames with a distance of 10 frames(Eye cameras have 200FPS).

Basically the eyeball can be computed continuously in a window with this method where we sample only once over all ellipses and select the 100 most different ones. The user of the software can specify the window size which is used to compute the eyeball. We also integrated the second approch from (Fuhl et al., 2020a), which allows computing an orthogonal vector from a single ellipse using a neural network, which can then be used to continuously adjust the eyeball model. The choice of the method to be used is selected via a parameter when calling Pistol.

3.4 Eye Movement Detection

For the detection of eye movements, we use the angles between the pupils and iris vectors as well as the difference of the eye-opening distance. For classification, we use a neural network with a hidden layer and 100 neurons, as well as the softmax loss. The network was trained on the annotated data of 12 subjects. In addition, we use the validity of the extracted eye fea-

Figure 6: Exemplary images of the marker detection. The yellow dot is the estimated center. **Zoom in to sea more details**.

Figure 7: Exemplary images of the 2D gaze estimation for the left (first 2) and right (second 2) eye, and exemplary images of the 3D gaze estimation for both eyes (last 4). The iris center, pupil center, iris vector, and pupil vector with neural networks and levenberg marquart polynomial fitting are drawn in different colors. In addition, each scene frame as ≈ 6 gaze points based on the frame rate of the eye camera. **Zoom in to sea more details**.

tures to classify errors.

Pistol supports also the fully convolutional approach from (Fuhl et al., 2020b) to segment the motion trajectories. This model is pretrained on TEyeD (Fuhl et al., 2021) and can be used to compute the eye movements for other eye trackers like the Dikablis, Emke GmbH, Look, Pupil, and many more, which are included in the TEyeD datset. This is necessary because there are significant differences in the eye trackers due to different camera placement and perspective distortion. We decided to provide a small seperate neural network since with the Pupil Invisible Eye Tracker, the classification is not a linear function, as eye movements near the nose have significantly smaller distances compared to areas of the eyes that are much closer to the camera. This is due to the different depth and of course the perspective distortion of the camera lens.

3.5 Marker Detection

Figure 6 shows results of our marker detection, where the center is shown as a yellow dot. Our marker detection must be able to detect the marker over different distances and should calculate the center as accurately as possible. To accomplish this in a reasonable time, we decided to use two DNNs. The first DNN gets the whole image scaled to 400×400 pixel. From this, the DNN generates a heatmap with a resolution of 50×50 pixel. The maxima in this heatmap are then used as the starting position for the second DNN, which extracts a 120×120 pixel area from the original image and performs landmark detection with validation from (Fuhl and Kasneci, 2019) here. With the validation signal, we again reject marker positions which are too inaccurate. The architecture of our models as well as the training details are in the supplementary material.

3.6 2 D & 3D Gaze Estimation

For the determination of the calibration function, we use the Levenberg Marquart optimization and neural networks with two hidden layers (50 and 20 neurons). The polyniomal and the neural network architecture can be changed with additional parameters of Pistol. It is also possible to use exponential functions instead of the polyniomal. This is due to the fact that Pistol supports different eye trackers, and both methods are able to learn more complex functions than simple polynomials. An example of this can already be found in the depth estimation, where the parameters are in the exponent of the function and thus cannot be determined via a direct computation method.

For the 2D eye tracking we fit a neural network and a polynomial with the Levenberg Marquart method to the pupil center, iris center, pupil vector, and iris vector. This is done separately and also for each eye independently. This means that people with an ocular incorrect eye position or only one functioning eye can also be measured. In the case of 3D eye tracking, we use the data from both eyes simultaneously. This means that we fit a neural network and a polynomial with the Levenberg Marquart method to both pupil centers, iris centers, pupil vectors, and iris vectors. Overall, the program therefore computes 8 gaze point estimators per eye (Total 16) and additionally 8 for both eyes in combination. All of these estimated gaze points can be used separately and are written to the csv files.

The selection of the calibration data is done in three steps. First, each scene image with a marker detection is assigned only one eye image for each eye. This assignment is done by the minimum distance of the time stamps. In the second step, we check that only scene images are used for which it is true that there are 5 valid marker detections in the preceding and following scene images. In the third step, we use the Levenberg Marquart method together with a polynomial. Based on the error of the gaze positions to the marker position, the best 90% are selected and the

Figure 8: Image of the marker area to depth estimation recording (Flipped by 90 degree). The brown markers on the floor have 50 cm distance to each other and the subject had to stand in front of the line for one measurement sample.

remaining 10% are discarded.

The training parameters for the neural networks are initial learning rate of 0.1, optimizer SGD, momentum 0.9, and weight decay 0.0005. Each network is trained 4 times for 2000 epochs, with batch size equal to all input data. After 2000 epochs, the learning rate is reduced by a factor of 10^{-1}. For the Leveberg Marquart method, we use the delta stop strategy with a factor of 1^{-10} and a search radius of 10.0. For the neural networks and the Leveberg Marquart method, all data are normalized. This means that the pupil and iris centers are divided by the resolution in x and y direction, as well as the eyeball centers for the pupil and iris vectors. The vectors themselves are already unit vectors.

3.7 Depth Estimation

For the determination of the depth, we have considered several methods. One is the fitting of polynomials and complex functions to the vectors of the pupil and the iris, as well as to the angle between both vectors. Also, we tried to determine the depth geometrically. In all cases there were significant errors, because the vectors are not linear to each other due to the perspective distortion of the camera, the nonlinear deep distribution of the eye image and also due to the head rotation. In the case of the head rotation, it comes to the eye rotation around the z axis as well as it also depends on the viewing angle to different eye positions comes, which are not linear to a straight view. Since neither neural networks nor complex functions worked for us, we decided to use the K nearest neighbor method (KNN) with $k = 2$. This method gave the best results in our experiments and a good accuracy (except for a few centimeters).

In order to determine the marker depth and thus obtain our calibration data for the KNN, we performed trial measurements with the marker and two people. The setup can be seen in Figure 8. The brown strokes on the floor are plotted at 50 cm intervals, and both subjects took measurements at each stroke. We then used the Matlab curve fitting toolbox to determine the best function and parameters for them.

$$\mathrm{d(A)} = a * \mathrm{A}^b + c \tag{2}$$

The best function can be seen in the Equation 2, where A is the area of the marker and $\mathrm{d}(A)$ is the depth in centimeters. The first thing to notice is that the parameter b is in the exponent, which makes determining the parameter not easy to solve directly. The parameters we determined for the function are $a = 13550.0f$, $b = -0.4656$, and $c = -18.02$. We use these parameters and function to determine the depth of the marker from the marker area using the fine detector.

4 EVALUATION

In this section we evaluate the different processing steps of the presented tool. If it was possible we compared it to other state of the art approaches. For the gaze estimation we compared to the Pupil Player but we want to mention again, that our tool has to be seen as an additional software to use with the Pupil Invisible and with other eye trackers too. It is not a competing product, nor do we wish to denigrate any software here.

For the evaluation of the gaze, we recorded five subjects who looked at the calibration marker at the beginning of the recording. Subsequently, the subjects moved freely inside and outside the university and at the end the calibration marker was viewed again. The observation of the calibration marker was at the beginning that the subjects stood close to the marker and then slowly moved away from the marker with head rotations. For the evaluation data, subjects were roughly five meters away initially and then moved towards the marker with head rotations. By doing this, we have depth information in the calibration as well as in the evaluation data (See section 3.7). The head rotation caused the marker to move in the scene image as well as eye rotations, making gaze determination much more difficult.

Each recording was approximately five minutes long, and each subject took three recordings. One shot could not be used because the subject had the marker in the scene image for a while without looking at it, resulting in a false calibration. Of course, Pistol filters out a certain amount of false data via the outlier selection, but this must not exceed 10% of the images with markers in the calibration phase.

Our system for the evaluation consists of an NVIDIA GTX 1050ti, an AMD Ryzen 9 3950X with 16 cores (where only one core was used in the evaluation), and 64 GB DDR4 RAM. The operating system

Table 1: We evaluated the segmentation quality with the mean intersection over union as well as the average euclidean pupil center error in comparison to other approaches from the literature on our annotated dataset (7 subjects for training and 5 for evaluation). In addition, we evaluated the iris landmark RMSE also with the euclidean distance.

Method	Pupil center RMSE (px)	Pupil mIoU	Iris LM RMSE (px)	Iris mIoU	Eyelid mIoU
Pistol	0.96	0.83	1.14	0.88	0.89
(Fuhl et al., 2016b)	8.92	0.49	-	-	-
(Fuhl et al., 2015)	11.13	0.35	-	-	-
(Fuhl et al., 2016a)	-	-	-	-	0.49
(Fuhl et al., 2017)	-	-	-	-	0.61

Table 2: Evaluation of the eye movement detection algorithm in pistol and compared to other state of the art approaches on our annotated data set (7 subjects for training and 5 for evaluation).

	Saccade		Fixation		Smooth Pursuit		Blink	
Method	Accuracy	mIoU	Accuracy	mIoU	Accuracy	mIoU	Accuracy	mIoU
Pistol	89%	0.81	96%	0.89	95%	0.89	86%	0.78
(Fuhl et al., 2018b)	90%	0.78	93%	0.85	90%	0.80	86%	0.77
(Fuhl et al., 2018a)	80%	0.69	89%	0.81	99%	0.85	66%	0.45
(Fuhl et al., 2020b)	92%	0.87	97%	0.90	97%	0.91	89%	0.83

is Windows 10 and the CUDA version used is 11.2, although Pistol also works with CUDA versions 10.X, 11.1, and 11.3.

4.1 Pupil, Iris, and Eyelid Detection

The results of the pupil, iris, and eyelid extraction networks can be seen in Table 1. As can be seen, the results of Pistol are very good compared to other methods, but the other approaches do not require training data. Of course, one could use much larger networks with even better results, but Pistol should be usable by everyone, so the resource consumption is also a very important property of the models. In the runtime section, you can see that our approach on an old GTX 1050ti with 4 GB Ram needs only 17.21 milliseconds per frame (Table 8) and also uses only a fraction of the GPU RAM. This makes it possible to use Pistol on older hardware.

4.2 Eye Movement Detection

Table 2 shows the accuracy of our eye movement classification compared to other methods from the literature. The best result is given by the method from (Fuhl et al., 2020b) which is also integrated in Pistol but for other eye trackers and pretrained on TEyeD. We did not select the model from (Fuhl et al., 2020b) as standard approach since it requires more computational ressources andthe fully convolutional approach is independent of the length of the input signal. Therefore, it can allocate a huge amount of ram, which can only be controlled by seperating the input vector, which would also impact the resulting accuracy. However, the current approach is better than HOV (Fuhl et al., 2018a) and as well as more accurate compared to threshold learning (Fuhl et al., 2018b) as used in the classical algorithms.

4.3 2 D & 3D Gaze Estimation

As already described in the introduction in the section Evaluation, we use different depths for calibration and evaluation as well as there are head and eye rotations. Table 3 shows the accuracy of our gaze determination with the different methods. As can be seen, the neural networks are significantly worse than the Levenberg-Marquardt fitting, which we suspect is due to the formulation of the individual neurons. These are single linear functions which are combined to approximate the seen area well and interpolate well between training data. However, in the case of extrapolation, neural networks are very poor. This will be the task of future research, and we hope to provide better networks in the future. Compared to the Pupil Player, we are significantly better on average, but this is also due to the fact that the Pupil Invisible Eye Tracker is only calibrated to one depth level and the software cannot use our markers.

Table 4 shows the frequency of detected viewpoints relative to all seen images from all shots. As can be seen, it is possible to determine significantly more gaze points with the Pupil Invisible than is provided for in the Pupil Player software. This is due to the fact that the eye cameras record 200 frames per second and the scene camera records 30 frames per second. Looking at the results, it is noticeable that the 3D gaze point can be calculated the least. This is due

Table 3: The average gaze estimation accuracy in pixels for all methods of Pistol for the five participants. LM stands for the Levenberg-Marquardt fitting, NN for the neural network fitting, PC for pupil center, IC for iris center, PV for the pupil vector, and IV for the iris vector. Since the Pupil Player outputs only up to two gaze points per scene image, we evaluated our approaches with all assigned gaze points to the scene image and only with the eye image with the lowest timestamp difference. The result of 2D left+right is the average of both gaze estimations. **Note: The Pupil Player uses only the one time calibration for one depth and therefore, the evaluation is not entirely fair**

Data	Method	Average Accuracy (px)							
		LMPC	LMIC	LMPV	LMIV	NNPC	NNIC	NNPV	NNIV
$\frac{\approx 6}{Image}$	Pistol 2D left	29.04	27.73	28.61	27.33	36.25	35.29	34.51	35.26
	Pistol 2D right	32.20	33.21	34.25	35.18	47.44	44.82	37.78	38.60
	Pistol 2D left+right	21.76	23.53	22.96	24.11	27.08	27.33	24.51	28.27
	Pistol 3D	20.20	21.24	19.61	20.45	28.78	30.22	26.86	27.91
$\frac{1}{Image}$	Pistol 2D left	24.84	23.61	24.42	23.26	32.68	31.51	30.53	31.34
	Pistol 2D right	28.01	29.12	29.93	31.11	43.89	40.81	33.83	34.62
	Pistol 2D left+right	18.54	20.21	19.62	20.88	24.22	24.54	21.66	25.23
	Pistol 3D	18.11	19.24	17.43	18.56	27.54	29.14	25.18	26.25
$\frac{\approx 2}{Image}$	PupilLabs	66.39							

Table 4: Since the Pupil Player outputs only one or two gaze points per scene image, we evaluated the total valid gaze points of all possible gaze points (Up to seven per scene image) as well as scene images with at least one gaze point both as percentage. For the evaluation, we used all recordings from all five participants.

Source	Left	Right	Left + Right	3D	Pupil Player
Valid gaze points	90.63%	93.71%	93.78%	88.14%	23.07%
At least 1 valid gaze point per scene image	91.77%	94.80%	96.89%	89.66%	99.32%

to the fact that both eyes must be valid here. However, due to the camera placement, the pupil is no longer visible in one of the eyes when looking from the side.

If we calculate the relative gaze points per scene image, i.e. that at least one viewpoint per scene image is sufficient, we see that here the Pupil Player determines significantly more gaze points than Pistol. For future versions of Pistol, this will be further improved, and we will also integrate an appearance based gaze estimation approach.

4.4 Pupil, Iris, and Eyelid Detection

The architecture of our models is described in table 5. With the four by three maximum interconnection blocks, our networks are between ResNet-18 and ResNet-34 (He et al., 2016) in size. For training, we used the AdamW optimizer (Loshchilov and Hutter, 2018) with parameters first momentum 0.9, second momentum 0.999, and weight decay 0.0001. The initial learning rate was set to 10^{-4} and reduced after 1000 epochs to 10^{-5} in which we trained the models for an additional 1000 epochs with a batch size of 10. We pretrained our model on the TEyeD (Fuhl et al., 2021) dataset and used all stages up to 8 as backbone model. Afterwards, we retrained the last layer on 6,000 annotated images from the pupil invisible with 3,000 additional annotated images for validation. The annotated data set contains 12 subjects and for the evaluation we split the training (7 subjects) and testing (5 subjects) data so that there was no subject from the training set in the testing set. For data augmentation we used random noise (0 *to* 0.2 multiplied by the amount of pixels), cropping (0.5 *to* 0.8 of the resolution in each direction), zooming (0.7 *to* 1.3 of the resolution in each direction), blur (1.0 *to* 1.2), intensity shift (-50 *to* 50 added to all pixels), reflection overlay (intensity: 0.4 *to* 0.8, blur: 1.0 *to* 1.2, shift: -0.2 *to* 0.2), rotation (-0.2 *to* 0.2), shift (-0.2 *to* 0.2 of the resolution in each direction), and occlusion blocks (Up to 10 occlusions and with size from 2 pixels up to 50 with a fixed random value for all pixels or random values for each pixel).

For other eye trackers (Dikablis, Emke GmbH, Look, Pupil, and many more), Pistol support also the feature extraction, if they are included in the TEyeD dataset (Fuhl et al., 2021). This can be specified as a parameter and the path to the videos of the eyes have to be specified. For those eye trackers, Pistol will use the last fully connected stage of the backbone network.

4.5 Marker Detection

Table 6 shows the results of our marker detection in the Euclidean distance to the marker landmarks, as mean intersection over union, which is important for the determination of the depth information, and the

Table 5: Shows the architecture of the DNNs used for pupil, iris, and eyelid landmark detection.

Level	Layer
Input	Gray scale image 192×192 *Center cropp and rescale for other resolutions.*
1	5×5 Convolution with depth 64
2	ReLu with tensor normalization
3	2×2 Max pooling
4	3 Maxium connection blocks with 64 layers and 2×2 average pooling integrated in the first block.
5	3 Maxium connection blocks with 128 layers and 2×2 average pooling integrated in the first block.
6	3 Maxium connection blocks with 256 layers and 2×2 average pooling integrated in the first block.
7	3 Maxium connection blocks with 512 layers and 2×2 average pooling integrated in the first block.
8	Fully connected with 1024 neurons and ReLu
Output	Fully connected with 48, 96, or 198 output neurons for pupil, iris, or eyelid landmarks in the same order.

Table 6: Results of the marker detection. The accuracy and the mean intersection over union is evaluated on our annotated data set. Accuracy is measured as average euclidean distance of the landmarks in pixel. The false detections are evaluated over all recorded videos and divided by the sum of all frames.

Method	Average landmark Accuracy (px)	mIoU	False detections
Coarse	7.12	0.60	3.26%
Fine	0.87	0.92	0.0%

false detections over all videos. As can be seen, the coarse detection of the markers is relatively error-prone with a false detection rate of 3.26% as well as very inaccurate. However, this is compensated by the fine detection, and it saves a lot of time to have to check only a few positions with the fine detection.

Table 7 shows the architectures of our DNNs. Both DNNs use tensor normalization (Fuhl, 2021b). The maximum connections (Fuhl, 2021a) and full distribution training (Fuhl, 2021b) were also used in the fine detector. To train the DNNs we annotated 1,250 images (1,000 for training and 250 for validation) where the coarse DNN used the entire image for training and the fine DNN used only the region of the marker in a 120×120 pixel area. The data set of the markers was recorded in five different rooms and on different walls. To ensure the generalization of our approach we used four rooms for training and one room for testing.

For data augmentation we used random noise ($0\,to\,0.2$ multiplied by the amount of pixels), cropping ($0.5\,to\,0.8$ of the resolution in each direction), zooming ($0.7\,to\,1.3$ of the resolution in each direction), blur ($1.0\,to\,1.2$), intesity shift ($-50\,to\,50$ added to all pixels), reflection overlay (intesity: $0.4\,to\,0.8$, blur: $1.0\,to\,1.2$, shift: $-0.2\,to\,0.2$), rotation ($-0.2\,to\,0.2$), shift ($-0.2\,to\,0.2$ of the resolution in each direction), and occlusion blocks (Up to 10 occlusions and with size from 2 pixels up to 50% of the image with a fixed random value for all pixels or random values for each pixel). This data augmentation was used for both DNNs.

For training of the fine detector, we used the AdamW optimizer (Loshchilov and Hutter, 2018) with parameters first momentum 0.9, second momentum 0.999, and weight decay 0.0001. The initial learning rate was set to 10^{-3} and reduced after 1000 epochs to 10^{-4} in which we trained the models for an additional 1000 epochs with a batch size of 10.

The coarse detector was trained with SGD (Rumelhart et al., 1986) and the parameter's momentum 0.9 and weight decay 0.0005. The initial learning rate was set to 10^{-1} and reduced after 1000 epochs to 10^{-2} in which we trained the models for an additional 1000 epochs with a batch size of 10.

4.6 Depth Estimation

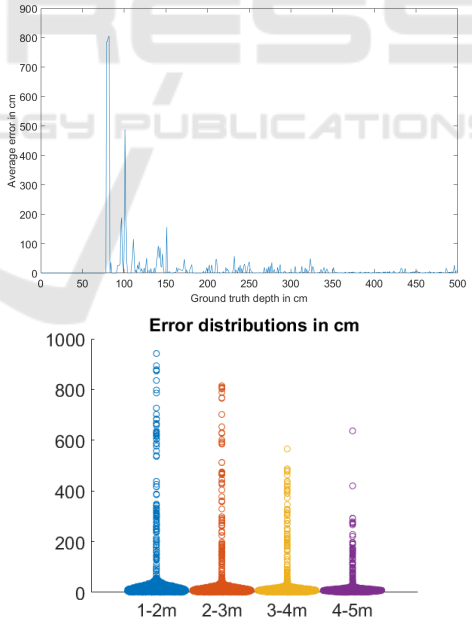

Figure 9: The average error per distance is shown on the right in cm. The left plot is the distribution of errors for the segments in 1 meter blocks. Both plots are computed on all recordings.

Figure 9 shows the accuracy of our depth estimation. On the left side, the mean values for all depths up to 5 meters are shown. As you can see, especially the near

Table 7: Shows the architectures of the DNNs used for the marker detection.

Level	Coarse detector	Fine detector
Input	Gray scale image 400×400	RGB image 120×120
1	5×5 Convolution with depth 32	5×5 Convolution with depth 32
2	ReLu with tensor normalization	ReLu with tensor normalization
3	2×2 Max pooling	2×2 Max pooling
4	5×5 Convolution with depth 64	1 Maxium connection block with 64 layers and 2×2 average pooling
5	ReLu with tensor normalization	1 Maxium connection block with 128 layers and 2×2 average pooling
6	3×3 Convolution with depth 1	1 Maxium connection block with 256 layers and 2×2 average pooling
7	4×4 Average pooling	Fully connected with 512 neurons and ReLu
8		Fully connected with 48 output neurons for the landmarks

Table 8: Runtime of each step of Pistol in milliseconds per image or one data instance, without the generation of the debug information and debug video, as average over 1000 data instances. The used GPU is a GTX 1050Ti and the CPU AMD Ryzen 9 3950X where only one core is used for the evaluation.

Step	Feat.	Ball	Open	Move	Sync	Mark	GazeLM	GazeNN	TrainLM	TrainNN
time	17.21	0.22	0.18	0.49	0.01	57.35	0.14	0.27	2958	6107

areas are highly error-prone, which is due to the fact that both eyes are very close to the nose, which makes the perspective of the camera placement difficult to detect and distorts the vectors. On the right side, you can see the frequencies in one meter blocks. Here it is clear that there are some outliers, but the most common estimates have an error below 50 cm. This clearly shows that the depth estimation especially for a lateral camera placement needs further research, and we hope to improve it in future versions.

4.7 Runtime

Table 8 shows the runtime of the individual steps of Pistol. The times are in milliseconds and include the use of the GPU for marker detection and feature extraction. In the first update, the use of Pistol will also be possible without GPU, and thus we can then also provide a version for MacOSX. The training for the gaze determination takes the most time, however, it is not only a data instance, but the complete training with all data. In the future, the marker detection will be accelerated as well, since it currently still takes too much time. However, it was important to us that the marker detection is very accurate and robust.

5 CONCLUSION

In this paper we have described in detail our software that allows to extract a variety of features from the eye as well as to perform 2D and 3D gaze estimation. We hope that the variety of extracted features will support research in many areas and that the software will be useful for many researchers. Future features of the software will be gesture recognition, scene analysis as well as scan path classification, which will hopefully make the software even more useful for research.

ACKNOWLEDGEMENTS

Daniel Weber is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645

REFERENCES

Arslan, O., Atik, O., and Kahraman, S. (2021). Eye tracking in usability of electronic chart display and information system. *The Journal of Navigation*, 74(3):594–604.

ASL (2021). Asl. https://est-kl.com/. [Online; accessed 04-November-2021].

Benjamins, J. S., Hessels, R. S., and Hooge, I. T. (2018). Gazecode: Open-source software for manual mapping of mobile eye-tracking data. In *Proceedings of the 2018 ACM symposium on eye tracking research & applications*, pages 1–4.

Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123.

Davis, R. and Sikorskii, A. (2020). Eye tracking analysis of visual cues during wayfinding in early stage alzheimer's disease. *Dementia and geriatric cognitive disorders*, 49(1):91–97.

Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470.

Economides, J. R., Adams, D. L., Jocson, C. M., and Horton, J. C. (2007). Ocular motor behavior in macaques with surgical exotropia. *Journal of neurophysiology*, 98(6):3411–3422.

Ergoneers (2021). Ergoneers. https://www.ergoneers.com. [Online; accessed 04-November-2021].

eyecomtec (2021). eyecomtec. https://eyecomtec.com. [Online; accessed 04-November-2021].

EyeTech (2021). Eyetech. http://www.eyetec.com/. [Online; accessed 04-November-2021].

Fuhl, W. (2021a). Maximum and leaky maximum propagation. *arXiv preprint arXiv:2105.10277*.

Fuhl, W. (2021b). Tensor normalization and full distribution training. *arXiv preprint arXiv:2109.02345*.

Fuhl, W., Castner, N., and Kasneci, E. (2018a). Histogram of oriented velocities for eye movement detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*.

Fuhl, W., Castner, N., and Kasneci, E. (2018b). Rule based learning for eye movement type detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*.

Fuhl, W., Gao, H., and Kasneci, E. (2020a). Neural networks for optical vector and eye ball parameter estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM.

Fuhl, W. and Kasneci, E. (2019). Learning to validate the quality of detected landmarks. In *International Conference on Machine Vision, ICMV*.

Fuhl, W., Kasneci, G., and Kasneci, E. (2021). Teyed: Over 20 million real-world eye images with pupil, eyelid, and iris 2d and 3d segmentations, 2d and 3d landmarks, 3d eyeball, gaze vector, and eye movement types. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 367–375. IEEE.

Fuhl, W., Kübler, T. C., Sippel, K., Rosenstiel, W., and Kasneci, E. (2015). Excuse: Robust pupil detection in real-world scenarios. In *16th International Conference on Computer Analysis of Images and Patterns (CAIP 2015)*.

Fuhl, W., Rong, Y., and Enkelejda, K. (2020b). Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction. In *Proceedings of the International Conference on Pattern Recognition*, pages 0–0.

Fuhl, W., Santini, T., Geisler, D., Kübler, T. C., Rosenstiel, W., and Kasneci, E. (2016a). Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct publication – PETMEI 2016*.

Fuhl, W., Santini, T., and Kasneci, E. (2017). Fast and robust eyelid outline and aperture detection in real-world scenarios. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*.

Fuhl, W., Santini, T., Kübler, T. C., and Kasneci, E. (2016b). Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications (ETRA)*, pages 123–130.

Gardony, A. L., Lindeman, R. W., and Brunyé, T. T. (2020). Eye-tracking for human-centered mixed reality: promises and challenges. In *Optical Architectures for Displays and Sensing in Augmented, Virtual, and Mixed Reality (AR, VR, MR)*, volume 11310, page 113100T. International Society for Optics and Photonics.

Hale, M. L. (2019). Eyestream: An open websocket-based middleware for serializing and streaming eye tracker event data from gazepoint gp3 hd research hardware. *Journal of Open Source Software*, 4(43):1620.

Hasselbring, W., Carr, L., Hettrick, S., Packer, H., and Tiropanis, T. (2020). Open source research software. *Computer*, 53(8):84–88.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

iMotions (2021). imotions. https://imotions.com/. [Online; accessed 04-November-2021].

Jermann, P., Nüssli, M.-A., and Li, W. (2010). Using dual eye-tracking to unveil coordination and expertise in collaborative tetris. *Proceedings of HCI 2010 24*, pages 36–44.

Jiang, Z., Chang, Y., and Liu, X. (2020). Design of software-defined gateway for industrial interconnection. *Journal of Industrial Information Integration*, 18:100130.

Jones, P. R. (2018). Myex: a matlab interface for the tobii eyex eye-tracker. *Journal of Open Research Software*, 6(1).

Joseph, A. W. and Murugesh, R. (2020). Potential eye tracking metrics and indicators to measure cognitive load in human-computer interaction research. *Journal of Scientific Research*, 64(1).

Lepekhin, A., Capo, D., Levina, A., Borremans, A., and Khasheva, Z. (2020). Adoption of industrie 4.0 technologies in the manufacturing companies in russia. In *Proceedings of the International Scientific Conference-Digital Transformation on Manufacturing, Infrastructure and Service*, pages 1–6.

Lev, A., Braw, Y., Elbaum, T., Wagner, M., and Rassovsky, Y. (2020). Eye tracking during a continuous performance test: Utility for assessing adhd patients. *Journal of Attention Disorders*, page 1087054720972786.

Li, D., Babcock, J., and Parkhurst, D. J. (2006). openeyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 95–100.

Liu, C., Chen, Y., Tai, L., Ye, H., Liu, M., and Shi, B. E. (2019). A gaze model improves autonomous driving. In *Proceedings of the 11th ACM symposium on eye tracking research & applications*, pages 1–5.

Loshchilov, I. and Hutter, F. (2018). Fixing weight decay regularization in adam.

Mao, R., Li, G., Hildre, H. P., and Zhang, H. (2021). A survey of eye tracking in automobile and aviation studies: Implications for eye-tracking studies in marine operations. *IEEE Transactions on Human-Machine Systems*, 51(2):87–98.

Meng, X., Du, R., and Varshney, A. (2020). Eye-dominance-guided foveated rendering. *IEEE transactions on visualization and computer graphics*, 26(5):1972–1980.

Nesaratnam, N., Thomas, P., and Vivian, A. (2017). Stepping into the virtual unknown: feasibility study of a virtual reality-based test of ocular misalignment. *Eye*, 31(10):1503–1506.

Niehorster, D. C., Hessels, R. S., and Benjamins, J. S. (2020). Glassesviewer: Open-source software for viewing and analyzing data from the tobii pro glasses 2 eye tracker. *Behavior Research Methods*, 52(3):1244–1253.

Oculus (2021). Oculus. https://www.oculus.com. [Online; accessed 04-November-2021].

Panchuk, D., Vine, S., and Vickers, J. N. (2015). Eye tracking methods in sport expertise. In *Routledge handbook of sport expertise*, pages 176–187. Routledge.

Park, J. and Yim, K. (2021). Technical survey on the real time eye-tracking pointing device as a smart medical equipment. *Smart Media Journal*, 10(1):9–15.

Pavisic, I. M., Pertzov, Y., Nicholas, J. M., O'Connor, A., Lu, K., Yong, K. X., Husain, M., Fox, N. C., and Crutch, S. J. (2021). Eye-tracking indices of impaired encoding of visual short-term memory in familial alzheimer's disease. *Scientific reports*, 11(1):1–14.

Phan, T. V. (2011). *Development of a custom application for the Tobii Eye Tracker*. PhD thesis, Hochschule für angewandte Wissenschaften Hamburg.

PupilLabs (2021). Pupillabs. https://pupil-labs.com/. [Online; accessed 04-November-2021].

Rakhmatulin, I. (2020). A review of the low-cost eye-tracking systems for 2010-2020. *arXiv preprint arXiv:2010.05480*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Santini, T., Fuhl, W., Geisler, D., and Kasneci, E. (2017). Eyerectoo: Open-source software for real-time pervasive head-mounted eye tracking. In *VISIGRAPP (6: VISAPP)*, pages 96–101.

Shinohara, Y., Currano, R., Ju, W., and Nishizaki, Y. (2017). Visual attention during simulated autonomous driving in the us and japan. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 144–153.

Snell, S., Bontempo, D., Celine, G., and Anthonappa, R. (2020). Assessment of medical practitioners' knowledge about paediatric oral diagnosis and gaze patterns using eye tracking technology. *International Journal of Paediatric Dentistry*.

SRResearch (2021). Srresearch. https://www.sr-research.com/. [Online; accessed 04-November-2021].

Taylor, M. P., Boxall, P., Chen, J. J., Xu, X., Liew, A., and Adeniji, A. (2020). Operator 4.0 or maker 1.0? exploring the implications of industrie 4.0 for innovation, safety and quality of work in small economies and enterprises. *Computers & industrial engineering*, 139:105486.

Tobii (2021). Tobii. https://www.tobii.com/. [Online; accessed 04-November-2021].

VIVE (2021). Vive. https://www.vive.com/. [Online; accessed 04-November-2021].

Voßkühler, A., Nordmeier, V., Kuchinke, L., and Jacobs, A. M. (2008). Ogama (open gaze and mouse analyzer): open-source software designed to analyze eye and mouse movements in slideshow study designs. *Behavior research methods*, 40(4):1150–1162.

Walton, D. R., Anjos, R. K. D., Friston, S., Swapp, D., Akşit, K., Steed, A., and Ritschel, T. (2021). Beyond blur: Real-time ventral metamers for foveated rendering. *ACM Transactions on Graphics (TOG)*, 40(4):1–14.

Zandi, B., Lode, M., Herzog, A., Sakas, G., and Khanh, T. Q. (2021). Pupilext: Flexible open-source platform for high-resolution pupillometry in vision research. *Frontiers in neuroscience*, 15.