Non-Intrusive Real Time Eye Tracking Using Facial Alignment for Assistive Technologies

C. Leblond-Menard and S. Achiche

***Abstract*— Most affordable eye tracking systems use either intrusive setup such as head-mounted cameras or use fixed cameras with infrared corneal reflections via illuminators. In the case of assistive technologies, using intrusive eye tracking systems can be a burden to wear for extended periods of time and infrared based solutions generally do not work in all environments, especially outside or inside if the sunlight reaches the space. Therefore, we propose an eye-tracking solution using state-of-the-art convolutional neural network face alignment algorithms that is both accurate and lightweight for assistive tasks such as selecting an object for use with assistive robotics arms. This solution uses a simple webcam for gaze and face position and pose estimation. We achieve a much faster computation time than the current state-of-the-art while maintaining comparable accuracy. This paves the way for accurate appearance-based gaze estimation even on mobile devices, giving an average error of around 4.5° on the MPIIGaze dataset (Zhang et al., 2019) and state-of-the-art average errors of 3.9° and 3.3° on the UTMultiview (Sugano et al., 2014) and GazeCapture (Krafka et al., 2016; Park et al., 2019) datasets respectively, while achieving a decrease in computation time of up to 91%.**

***Index Terms*— Gaze estimation, human computer interaction, assistive technology, neural networks.**

I. Introduction

ASSISTIVE robotic arms (ARA) are known to be greatly beneficial to people with upper-limb disabilities [5], [6], [7], [8], and [9]. Therefore, great efforts are put into the human-computer interfaces (HCI) to control these robots. Indeed, most commercially available ARAs come with joystick-based control devices, but those require significant amount of time to learn how to be used and to accomplish simple tasks, even for typically developed users [10]. Even though control filters have been implemented to help reduce movements caused by involuntary jerks from the user [5], the joystick based HCIs are generally much more frustrating to use than more automated HCI, depending on the user's impairment [6]. In some cases, potential users might have specific disabilities which make the use of a joystick via their hands impossible. As pointed by [11], the limits of the control interfaces currently available for commercial ARAs creates a situation where the people that should benefit the most of the ARAs are unable to control them due to their severe motor disability. Thus, the research work presented in [6] reports that a vision-based interface with autonomous path planning would lead to a significant improvement in user workload in grasping and pick-and-place tasks. Indeed, the user gaze in grasping tasks has been proven to be a large part of predicting intent and point of contact [12].

As such, current trends include using a camera, either grayscale/color or combined with a depth sensor, to detect objects on the scene presented in front of the user and a HCI to make decisions on what and how to manipulate these objects [13], [14], [15], [16], [17], [18]. Recent work has promoted the use of eye tracking as a HCI as demonstrated by [12], [14], [19], [20], [21], [22], [23], [24], [25], [26], and [27], albeit with significant limitations that will be further discussed in the next section.

A. Eye Tracking for Assistive Robotic Arms Control

Several extensive literature reviews are available that describe some of the current and previous state-of-the-art methods of eye tracking and gaze estimation, with a recent example being [28]. An even newer literature review has recently been published with comparative results in terms of angular gaze accuracy for deep learning based methods [29]. An overview of all the available methods is out of the scope of this paper.

One of the most popular and commercially available eye tracking techniques relies on using infrared (IR) corneal reflection aptly named *Pupil Center Corneal Reflection* method (PCCR). This method relies on an IR illuminator producing a distinctive reflection on the user eye's cornea for which the position can be compared with the more easily identifiable iris center (because it usually appears as a deep black circle). While this method produces accurate results [28], its reliance on IR reflection makes it particularly prone to error when exposed to sunlight, as [14] experienced. Otherwise, other accurate methods rely on the user wearing a head-mounted eye tracker or other intrusive systems when used in controlling ARAs.

Other methods have been recently proposed that make use of a single monocular camera to estimate the gaze direction [4], [30], [31], [32], [33], but those are, in our experience, either imprecise or ill-suited for embedded systems due to

Manuscript received 1 June 2022; revised 17 October 2022 and 15 November 2022; accepted 2 January 2023. Date of publication 17 January 2023; date of current version 3 February 2023. *(Corresponding author: C. Leblond-Menard.)*

The authors are with the Department of Mechanical Engineering, Polytechnique Montreal, Montreal, QC H3T 1J4, Canada (e-mail: cedric.leblond-menard@polymtl.ca; sofiane.achiche@polymtl.ca).

Digital Object Identifier 10.1109/TNSRE.2023.3236886

their computing performance requirements, especially in the case where no person-specific calibration frames are used.

Therefore, in this paper a novel non-intrusive real-time gaze estimation technique for use with ARAs that work reliably both indoors and outdoors while requiring as low a power as necessary when running on embedded systems without requiring person-specific calibration. The performance of the developed method is then compared to openly available state-of-the-art algorithms.

B. Objectives

Given the context of using ARAs, we therefore have the following research objective:

> Develop an open-source real-time gaze estimation model which can run on mobile and embedded devices in both indoor and outdoor environment using a single camera and without person-specific calibration.

Indeed, if this objective is completed, the resulting model could be used in systems such as [13] and [14] to provide a more versatile solution at a low cost.

C. Contributions

The contributions of this paper are as follows:

- We demonstrate that using state-of-the-art real-time face detection and iris alignment convolutional neural network (CNN) based algorithms, a gaze estimation system can provide an increased accuracy in estimating the gaze angles comparable to the current state-of-the-art while requiring no person-specific calibration and having a low performance requirement such as running in real-time on mobile devices.
- We bring forth a new framework of eye tracking tailored for robotics controlled that works reliably both indoors and outdoors while providing real-time performance, over 20 frames per second even on a single CPU core, and without being intrusive to the user. This model also outputs facial landmarks useful for assistive tasks as well.

II. Related Work

As introduced earlier, several methods that make use of a single camera aimed at the user's face have been recently described in the literature. As a comparison basis, we take particular interest in gaze estimation datasets that include full images of the user's face, as this allows for a more realistic scenario in the context of assistive devices as the face position must be located with respect to the camera to give an accurate estimation of the gaze angles.

One of the most widely used datasets for gaze estimation proposing full facial images is MPIIFaceGaze [34]. MPIIFaceGaze is a modified subset of the widely used MPIIGaze dataset [35] that includes complete faces instead of the eye region only found in the original dataset. It consists of 37,667 images and their corresponding gaze data taken across 15 participants. A recent paper described a new dataset called RT-GENE [31], but this dataset corresponds to scenarios where the user is not the main object of the picture and thus the faces are generally far from the camera. As this does not represent a typical scenario for gaze estimation control of ARAs, this dataset was not used in the context of this paper.

On the MPIIGaze dataset, the currently most accurate method available in the literature is called FAZE [4] and uses a combination of deep learning models to estimate the gaze direction and head pose of the user. This method implements a metalearned model to generate the weights of a gaze estimation network from only a few calibration frames (32 and less). As reported by the authors [4], the accuracy when using no calibration frames is 5.23°. The authors note that a real-time demo is available while providing no further description of computational performance.

Furthermore, the authors of the RT-GENE dataset proposed a gaze estimation model based on first extracting the facial landmarks using Multi-Task Cascaded Convolutional Networks [36] then correcting for the face image for the pose perspective against averaged face pose landmarks. The corrected eye patches are then extracted from the image and fed to VGG-16 networks, in an ensemble or not [37], one for each eye patch, to estimate the gaze direction. The reported gaze estimation error is 4.3° on MPIIGaze using an ensemble of 4 models.

Most other recent deep learning-based methods have achieved an angular accuracy ranging from 4.1° to 7.3°, as described in [29].

In 2019, a gaze estimation dataset named GazeCapture generated from phone and tablet gaze estimation trials was used [3] for angular gaze estimation [4]. While the main use of the dataset is to train gaze estimation models for estimating a gaze point on a mobile device's screen, thus the accuracy values generally given are in pixels, it is possible to use the dataset for gaze angles estimation by converting the pixel points to 3D points and then gaze angles, as is done in [4].

Moreover, another dataset that is widely used for gaze estimation tasks is the UTMultiview dataset [2]. This dataset offers a larger variety of head poses and gaze directions by using synthesized images from reconstructed faces using an array of cameras.

III. Methodology

A. Workflow and Implementation Details

The overall workflow proposed here can be separated in four distinct sections, which will be described here. These sections are:

1) Face detection
2) Head pose correction
3) Eye patch extraction
4) Iris detection and gaze estimation

As such, the input of the workflow is an image containing the face of the user. This image should be a color image and contains enough details to distinguish the iris from the sclera and eye contour. The output is composed of the gaze angles, pitch for the up-down eye motion and yaw for the left-right motion, and the gaze origin is the center point between the eyes. To ensure the accessbility of this gaze estimation method, we ensure that no step requires person-specific calibration, rather relying on existing large dataset on which to train on once prior to using this model.

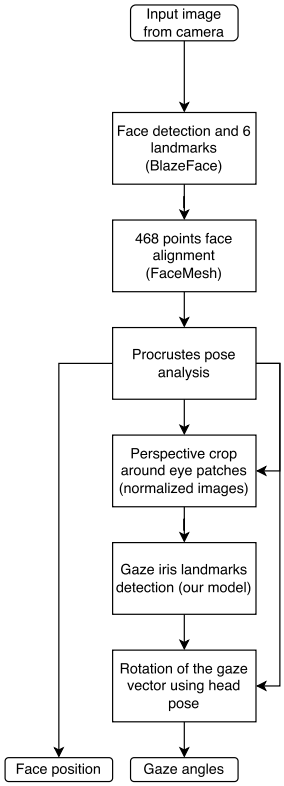

Fig. 1. Our workflow is mostly sequential, with the head pose being used to normalize the eye patch images and correct the estimated gaze.

Initially, a face detection algorithm is run on the image to detect the most prominent face in it. From that detection, we then must extract the location of the eyes, which is generally done through facial landmark extraction (generally named facial alignment). Given the position of the eyes, we can then extract a pair of image patches of each eye. These image patches are then used as the input of the gaze estimation algorithm, which outputs the gaze angles (direction). Since the position of the user's head should be known to infer the gaze point of gaze from the gaze direction, the facial landmarks can be used to compute the origin of the gaze vector.

Moreover, to reduce the influence of the head pose on the variability of the input images, we removed the roll component of the head pose and used the roll-removed head pose yaw and pitch angles as input to our model. Indeed, the world coordinates gaze direction $\overrightarrow{g}$ is a function of the head pose matrix H and the eye gaze direction with respect to the face $\overrightarrow{g_e}$:

$$\overrightarrow{g} = H\ \overrightarrow{g_e}$$

Note that the vectors and matrices given here are in homogeneous coordinates. Since $\overrightarrow{g}$ and $\overrightarrow{g_e}$ are directions, the last component of the homogeneous vectors is 0. The method of obtaining the head pose H and applying the perspective transform will be discussed in Section III-A.2.

1) Face Detection: In general, the approximate face location must be known beforehand to crop the image around the face center and perform the facial feature alignment and iris detection. Once a general location of the face is known, it is possible to skip the face detection step and use the general region of the previously detected face to perform the facial alignment step, especially if a confidence score is available from the alignment step which is true in our case. This thus saves time in processing a frame in a continuous video stream. A simple static threshold can be used such that if the confidence score falls below a fixed value, the face detection step is run again. For every face alignment step, we compute the center position of the face and check how far it moved from the last frame, and then re-centering the face-cropped image by moving the center by how far the face alignment center moved.

It should be noted that in the comparative results of this paper, this strategy is not used, since the datasets used for comparision are not videos, but rather distinct images and as such this strategy cannot work.

One of the first successful real-time face tracking algorithm was described by Viola and Jones using Haar-like cascades [38] and is still used. More precise and stable real-time algorithms have been proposed afterward, including histogram of oriented gradients (HOG) with support vector machines (SVM) and linear binary pattern (LBP) cascades, with HOG being generally the most accurate of these methods [39]. Recently, breakthroughs in small optimized CNNs led to very accurate face detection yet real-time performance on embedded computers and portable devices. One such state-of-the-art model, named BlazeFace, was developed and trained by Google Research with very high precision and fast inference time on mobile devices [40]. Based on MobileNetV1/V2 [41], [42], the architecture implements further optimization including increasing the receptive field size by using 5×5 kernels over 3×3 which are cheaper than adding layers with more computationally expensive pointwise convolutions.

Whereas our initial work was based on a modified HOG algorithm [43] parallelized and ported to CUDA, the recently release BlazeFace algorithm performs about as fast in our experience while providing increased accuracy comparable to state-of-the-art real-time models according to [40]. This is thus the chosen algorithm for our workflow. Furthermore, this algorithm outputs the location of 6 facial landmarks (the two eye centers, the two ear centers, the nose center, and the mouth center) which we can use to compute and correct for the head roll by assuming the left and right landmarks should be on a horizontal line.

2) Head Pose Correction: As stated previously, we can correct for the head pose by using the detected facial landmarks and applying a perspective transform to the eye image patches as a form of normalization, as was suggested in [1]. This allows the network to train on estimating the gaze direction without having to directly account for the head pose variation in the image. The pose normalization technique we used is based on the one described in [31].

As such, this requires several facial landmark points to be known on which to find the optimal series of rotation to align the actual head pose with a reference (sometimes called "canonical") head pose.

An effective real-time method of facial alignment recently published is called FaceMesh [44] and is part of the same augmented reality framework as BlazeFace developed by Google. It uses a straightforward residual neural network that

outputs a set of heatmaps for each facial feature (in this case 468 points) on which subpixel maximum estimation is done to find the input image location of the landmarks. This aligner is trained to not only output the 2D pixel location of each landmark, but also a depth value associated with each landmark that corresponds to the difference with the average depth of the face, while keeping the same scale (or aspect ratio) as the horizontal coordinates.

The face alignment model is learned from deforming a reference model with dimensions given in centimeters but projected to the image according to the camera model i.e., the camera matrix and distortion coefficients. Therefore, we can iteratively find the projected 3D transformations that best fit the transformation from the reference model and the computed landmarks points through a Procrustes analysis, also known as an orthogonal Procrustes problem, as used by the team at [45] and described in [46].

Using this method, we thus compute the head pose with respect to the camera, the previously described H. We can then find the gaze vector origin which is assumed to be the eye landmarks center point by using the canonical model's eye center $\overrightarrow{e_c}$ and head pose matrix H:

$$\overrightarrow{e} = H\overrightarrow{e_c}$$

Here, $\overrightarrow{e}$ is the eye center position with respect to the camera. Since the vectors here are positions, their last component is 1.

Given the direction to $\overrightarrow{e}$ and head pose H, we can find the perspective transformation matrix W that derotates the image as to make the head appear upright and that aligns the camera view axis with the eye position e while reprojecting the image as if the eye was at a new distance $\|e\|^*$ and a new camera focal length l^*. This methods was first proposed by [2] and then revisited by [47], the latter being the normalization technique used in this paper. By choosing the ratio between the new eye distance $\|e\|^*$, the new camera focal length l^* and the width and height of the normalized image, the resulting normalized image will have the de-rolled eye centered in the image with a specific and constant scale. In our case, the values used for $\|e\|^*$ and l^* are 600 millimeters are 650 pixels respectively. The distance $\|e\|^*$ of 600 millimeters is generally suggested by the literature [2] whereas the focal length l^* is generally found by trial and error until a value is found that has the right scaling ratio as to make the eye appear the choosen size for the model and input image resolution.

This removes parts of the perspective variance due to the head pose roll and camera parameters from the image and thus reduces the complexity of the problem to be learned by the gaze estimation model [1], [4], [31].

3) Eye Patch Extraction: From the facial landmarks, eye image patches must be extracted for the iris detection and gaze estimation part of our workflow as seen on Fig. 2. To do so, we use the average position in 3D of the eye from the previous step perform the perspective transformation normalization described in [31]. This method corrects for the head pose by warping the image as to align the eye patch normal vector to the vector between the camera's position and the eye position in 3D and re-projecting the image at a constant distance. This constant distance is set as to allow for a 25% margin on each side of the eye patch image, as is required by the iris detection and gaze estimation model that will be discussed in the following section.

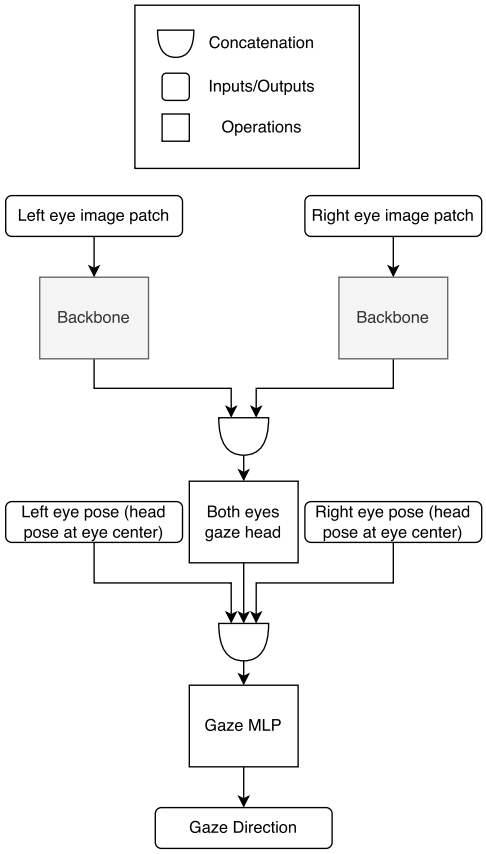

Fig. 2. A simplified overview of our model. Blocks with a darker fill represent parts of the model reused from the trained iris landmarks model from Mediapipe.

4) Iris Detection and Gaze Estimation: It is intuitive to think that the eye region landmarks and the iris center contain enough information about the user's gaze to estimate its direction. Indeed, the literature seems to point to a clear consensus on the subject [30], [31], [33], [48].

As such, we start from a model made to obtain the position of the iris given a cropped image of the eye on the right side from the user's perspective. For this model, we used as basis the very lightweight (few learned parameters) method recently described in [49] to align the eye region and find the iris center and contour landmarks.

While our initial work was based on an explicit relation in position between the eye region features (namely the lower sclera contour) and the iris center, where the vector component in the difference in position was used as the input of a basic polynomial regression or SVR regression to keep the computational burden to a minimum. This unfortunately in our experience lead to poor accuracy when compared to the state of the art, even when corrected for the head pose.

Considering the architecture of the aforementioned CNN model, we can infer that there is a lot of information encoding the position of the eye region landmarks and iris in the latent space between the backbone and the landmark heads (see Fig. 2).

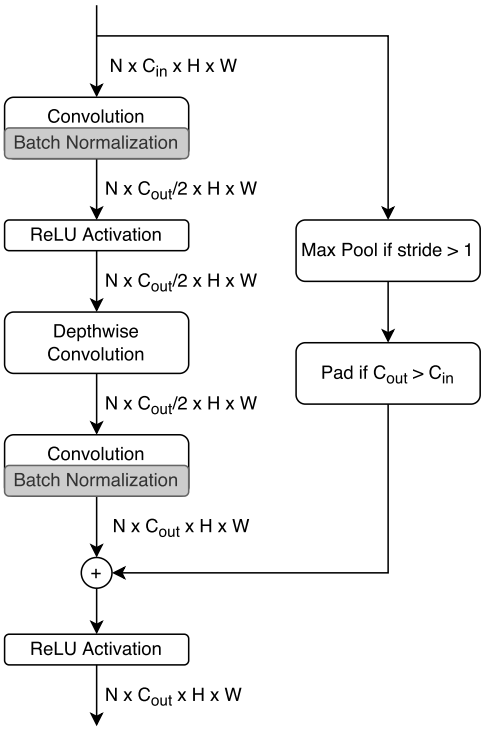

Fig. 3. The Irisblock is a variation on the BlazeFace blocks. They are a form of Residual blocks, but using a depthwise convolution to reduce the number of operations. The batch normalization layers used during training are in darker shade.

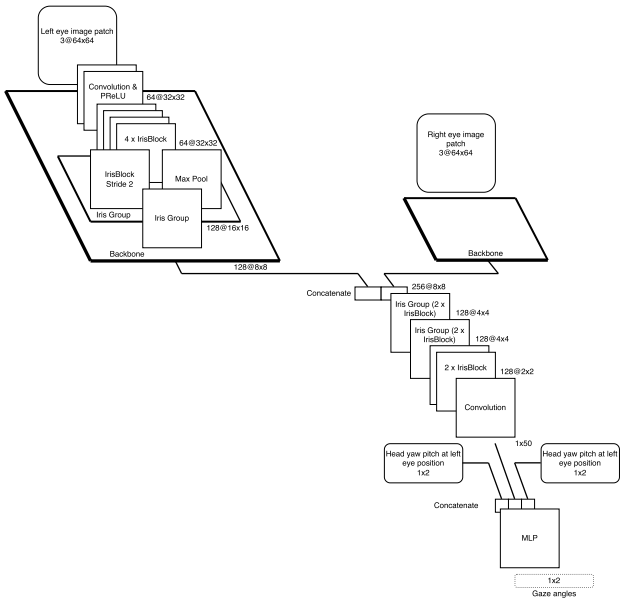

Fig. 4. Our gaze estimation and iris landmarks recognition model. The tensor dimensions are given between each operation. For the sake of simplicity, some operations implementations are not duplicated but are shown as white boxes.

Therefore, we ended up modifying the iris detection model by getting rid of the iris and eye landmarks splits of the original model and adding a new split to estimate the gaze direction. By using the initial model's weight and biases of the backbone, we can ensure that the model has a headstart in learning eye-specific encodings. We then concatenate the computed features of the head with the eye poses (face pose at each eye) as an input to a final, fully connected set of layers. The final output if this set of layers, a multi-layer perceptron (MLP), is thus the estimated gaze angle.

The backbone and split are made of Irisblocks [49] which are themselves based on Blazeblocks [40] which are themselves a modification of Residual blocks [50] where a convolution and activate layer is followed by a depthwise convolution followed by a normal convolution while the residual feed forward is an identity function or a max pool layer when the stride of the convolution layers lead to a reduction in the width and height dimensions. The sum of the convolutions and the residual are then passed through an activation function, PReLU in this case. In the specific case where the number of channels of the output of the block is higher than the number of input channels, the residual is padded to ensure the same number of channels when it is added to the output of the convolutions. See Figure 3.

Furthermore, we can double the model to make use of images from both eyes using a horizontally flipped image of the left eye and concatenate the feature vectors of each model split as the input for the gaze estimation head to improve the accuracy at the cost of increased computational complexity, as seen in Fig. 2 and 4.

By reusing the same backbone weights but horizontally flipping the left eye image, we found that our model often achieves better generalization, i.e. lower evaluation error.

Also, as suggested in [31], the head pose angles obtained from the Procrustes analysis and used for image normalization can be added as input to the model to account for some change in appearance of the eye due to the pose differences between frames. As such, we directly concatenate the head pose angles to the other features before the final fully connected layers.

5) Other Implementation Details: All the used deep learning models were implemented in PyTorch [51] and PyTorch Lightning [52], allowing them to run on both the CPU and GPU interchangeably. As such, our method is written in Python and PyTorch and makes the use of OpenCV [53] just for some basic image processing functions.

Although we train our modified version of the iris landmarks localization model proposed in [49], our workflow uses both the BlazeFace [40] model and the FaceMesh [44] model as trained by and available in Mediapipe [45], albeit with the models converted to PyTorch while keeping the same parameter values. The original model was implemented for TensorFlow-Lite [54] inference.

B. Training and Validation

Given our gaze estimation method is based on the iris landmarks localization model from [49], we start by re-using the model's parameters that correspond to the unmodified model backbone as the backbone of our model. This allows us to re-use the learned feature extraction for the eye and iris landmarks to help convergence of the optimization.

Training is done on the MPIIGaze [1] dataset by following the splits given in [4] and [31]. As such, for the MPIIGaze dataset, we use a leave-one-out approach for the test set.

We trained our model using the Ranger21 [55] optimizer, which implements recent progress on Rectified Adam [56] and

other methods that rely less on fine-tuning hyperparameters. The batch size is 32, the learning rate is set to the default 0.001 and the β_1 and β_2 parameters are set to 0.8 and 0.7 respectively to slow down the training speed. We also use a weight decay penalty of 0.001. The loss function is the mean square errors between the output gaze angles and the ground truth gaze angles as given by the direction between the eye center points (between the eyes) and the gaze targets of the dataset.

C. Performance Comparison Methodology

In order to allow for a common-ground comparison between the different algorithms tested, the PyTorch version of the tested methods was preferred if other implementations were available. Furthermore, we did not make use of optimized runtimes such as TensorRT that would provide an increase in inference speed for the same reason. All algorithms are run from Python and the inference time is taken from the start of the computation, just after the image frame is loaded in memory, to the end of the gaze estimation after the gaze angles are obtained, without any user interface processing or drawing functions being run.

The error metric e used is the arc cosine of the cosine similarity measure, given by computing the angle between the real gaze direction $\overrightarrow{g}$ given by the annotations of the dataset and the estimated gaze direction $\overrightarrow{g}_e$ given by the gaze angles, defined as:

$$e = \cos^{-1} \frac{|\vec{g} \cdot \vec{g}_e|}{\|\vec{g}\| \|\vec{g}_e\|}$$

This metric has interesting properties for comparison, as it is always positive and thus the average corresponds to the angular error between the directions, as opposed to the mean angular error in the pitch and yaw angles separately.

IV. Results

The results are presented in three distinct tables. Table I presents the results as either given by the original authors or in our case using the dataset supplied head pose information for the MPIIGaze [1] dataset.

Table II presents the results for the GazeCapture dataset [3] using the annotations and evaluation split defined by [4].

Table III on the other hand contains the results for the UTMultiview dataset [2] using the 3-fold evaluation split as used in [2] and [57]. In our case, since our model uses both eyes images as input, we rendered the same head poses for each sample (−36 to 36 °), but rendering both eyes instead of one at a time. This needs to be done as the left eye and right eye images of the pre-rendered images given with the dataset are not matched to one another.

Table IV contains the results we obtained running the implementation of the compared algorithms as distributed by the original authors but modified to remove all drawing and interface-related parts of the original code to allow for a better comparison in performance. All implementations are available on the authors Github repositories [57], [58]. The trained weights used are those made available by the original authors and we use a 7500 images subset of the MPIIFaceGaze dataset [34] to evaluate on.

TABLE I
ACCURACY FOR THE MPIIGAZE DATASET

Method	Accuracy
FAZE (no calibration) [4]	5.2°*
RT-GENE (4-ensemble) [31]	**4.3 ±0.9°**
GazeIrisLandmarks (ours)	4.5 ±1.2°

*The standard deviation is not provided in the article.

TABLE II
ACCURACY FOR THE GAZECAPTURE DATASET

Method	Accuracy
FAZE (no calibration) [4]	3.5°
GazeIrisLandmarks (ours)	**3.3°**

As this dataset has a single evaluation set and not a leave-one-person-out evaluation, no standard deviation figures are provided. No standard deviation figures were provided in the FAZE article as well [4].

TABLE III
ACCURACY FOR THE UTMULTIVIEW DATASET

Method	Accuracy
RT-GENE (4-ensemble) [31]	5.1 ±0.2 °
GazeIrisLandmarks (ours)	**3.9 ±0.5** °

TABLE IV
PERFORMANCE TIMINGS (FULL DETECTION AND GAZE ESTIMATION)

Method	Frame Times (milliseconds)		
	GPU	CPU	SC
FAZE (no calibration) [4]	228	254	431
RT-GENE (4-ensemble) [31]	36.4	255	323
GazeIrisLandmarks (ours)	**22.2**	**38.9**	**38.3**

Frame times are given from the moment after the image frame is available to the moment after which the gaze vector is available. GPU corresponds to a run with GPU acceleration, CPU using cores but without GPU acceleration and SC is using a single CPU core. Results are given without using the annotations but instead relying on the full workflow as made accessible by the authors to compute the estimated gaze direction and origin.

We decided not to include accuracy figures of this evaluation since it is not clear on what dataset the provided weights were trained on for the FAZE model. [4]

In Table IV The time performance metric (inference time) is given as an average in three scenarios: using GPU acceleration, without GPU acceleration but using all CPU cores and without GPU acceleration and using a single CPU core. This gives a better idea of the available performance in power constrained scenarios such as in mobile applications and embedded systems.

V. Discussion

A. Interpretation

As the results show, our model reached accuracy levels comparable to the current state-of-the-art while being significantly faster, especially when running only on the CPU. Given the original goal of using this gaze estimation model for assistive technologies such as controlling an assistive robotic arm directly mounted on a motorized wheelchair, this performance improvement is significant. We thus achieve a reduction of computing time of 39% against the fastest method in the worst-case scenario and up to 91% against the slowest in the best-case scenario.

Moreover, as we can see from the frame times, our GazeIrisLandmarks model requires lower computation time due to the low number of parameters. In fact, as can be seen from the multi-core and single core frame times, the

overhead of splitting the task on multiple cores makes the overall computation almost as slow as running it on a single core. This is confirmed by the model size used, with ours being 3.8 megabytes, whereas the RT-GENE 4 models ensemble is 1312 megabytes in total and the FAZE model is 27.5 megabytes.

As for the accuracy of our model, Tables I, II and III show that our model can reach accuracy figures comparable to the current state-of-the-art, being better in certain datasets and slightly worse in others.

B. Limitations and Further Work

Throughout our research, we have found that performance and computational requirements are rarely discussed extensively, let alone analyzed quantitatively, especially when comparing gaze estimation techniques. There is currently no benchmark or standardized way of comparing computational requirements of gaze estimation methods. This thus requires making the original model code available online for comparisons to be made and thus limits the number of models that can be compared.

Indeed, because performance numbers such as inference time depend on the hardward on which the numbers are produced, comparison must often be relative, where performance numbers for all compared methods must be run on the same computer as is the case for this paper.

It should also be noted that advances in technologies and new dedicated hardware, such as inference hardware now found in some smartphones could favor some algorithms over others, which is a limitation of the comparison method used in this paper.

Moreover, as has been demonstrated by [4], deep learning models can be adapted to leverage metalearning frameworks to improve accuracy of models by using calibration with very few samples. Given the context of assistive technologies, very long calibration procedures might be off-putting or just not possible but acquiring just a few calibration samples might be a good way to improve person specific accuracy and is therefore considered as a possible path of improvement. The drawback of such an approach is the added hyperparameter tuning required on top of the very high training computational requirements since higher order gradients are required [4].

Finally, the work presented in this paper focuses on appearance-based gaze estimation methods using singular cameras. Other methods exist, such as infrared reflections-based PCCR methods [28] which are often found in commercial eye trackers or calibration-based methods [33].

Some examples of the limitations of appearance-based gaze estimation methods without calibration includes lower accuracy when compared to PCCR methods and training dataset biases, where for example the ethnicity of the subjects in the training dataset limits the attainable accuracy with some users [59].

Furthermore, the domain of gaze and head pose angles for which an appearance-based model achieves accurate results is in our experience limited by the domain on which it is trained. As such, use cases for appearance-based models should take into consideration the training dataset's gaze and head pose angles domain and the typical use case angles to ensure they remain within the dataset's domain.

VI. Conclusion

We have shown that leveraging existing very small pre-trained models for eye region landmarks recognition and modifying the structure to access the latent information within the model can lead to accuracy comparable to the current state-of-the-art while significantly reducing the computational requirements. While we achieve a 4.5° average error which is similar to the current state-of-the-art for appearance based gaze angle estimation for the MPIIGaze dataset [29], we see a decrease in computation time ranging from 39% up to 91% against the current appearance-based gaze estimation methods publicly available. We also achieve a state-of-the-art 3.9° average error on the UTMultiview dataset [2] and 3.3° error on the GazeCapture dataset [3], [4].

This also allows the model to produce face and eye landmarks that can be used for other vision-based assistive tasks as well without needing further computation because these features are computed as part of the gaze estimation workflow. Furthermore, we show that a holistic approach describing a complete workflow such as proposed in this paper leads to improved accuracy when translated to real-world scenarios due to the high dependency of certain models to accurate and manually obtained annotations.

An implementation of our method, including a trained model, training description and real-time demo is available at: *https://github.com/cedriclmenard/fastgaze*

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "MPIIGaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 162–175, Jan. 2019.

[2] Y. Sugano, Y. Matsushita, and Y. Sato, "Learning-by-synthesis for appearance-based 3D gaze estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1821–1828.

[3] K. Krafka et al., "Eye tracking for everyone," Jun. 2016, *arXiv:1606.05814*.

[4] S. Park, S. De Mello, P. Molchanov, U. Iqbal, O. Hilliges, and J. Kautz, "Few-shot adaptive gaze estimation," Oct. 2019, *arXiv:1905.01941*.

[5] A. Campeau-Lecours et al., "JACO assistive robotic device: Empowering people with disabilities through innovative algorithms," in *Proc. Annu. Conf. Rehabil. Eng. Assistive Technol. Soc. North Amer. (RESNA)*, Arlington, VA, USA, 2016. [Online]. Available: https://corpus.ulaval.ca/entities/publication/4d72ec44-c987-2b71-e053-2528090a90b1 and https://www.resna.org/sites/default/files/conference/2016/other/campeau_lecours.html

[6] C.-S. Chung, H. Wang, and R. A. Cooper, "Functional assessment and performance evaluation for assistive robotic manipulators: Literature review," *J. Spinal Cord Med.*, vol. 36, no. 4, pp. 273–289, Jul. 2013.

[7] S. S. Groothuis, S. Stramigioli, and R. Carloni, "Lending a helping hand: Toward novel assistive robotic arms," *IEEE Robot. Autom. Mag.*, vol. 20, no. 1, pp. 20–29, Mar. 2013.

[8] V. Maheu, J. Frappier, P. S. Archambault, and F. Routhier, "Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities," in *Proc. IEEE Int. Conf. Rehabil. Robot.*, Jun. 2011, pp. 1–5.

[9] F. Routhier, P. S. Archambault, M.-C. Cyr, V. Maheu, M. Lemay, and I. Gélinas, "Benefits of JACO robotic arm on independent living and social participation: An exploratory study," in *Proc. RESNA*, Indianapolis, IN, USA, 2014. [Online]. Available: https://www.resna.org/sites/default/files/conference/2014/Robotics/Routhier.html

[10] S. Allin, E. Eckel, H. Markham, and B. R. Brewer, "Recent trends in the development and evaluation of assistive robotic manipulation devices," *Phys. Med. Rehabil. Clinics North Amer.*, vol. 21, no. 1, pp. 59–77, Feb. 2010.

[11] B. D. Argall, "Autonomy in rehabilitation robotics: An intersection," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 1, no. 1, pp. 441–463, May 2018.

[12] M. Karrenbach, D. Boe, A. Sie, R. Bennett, and E. Rombokas, "Improving automatic control of upper-limb prosthesis wrists using gaze-centered eye tracking and deep learning," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 340–349, 2022.

[13] C. Bousquet-Jette, S. Achiche, D. Beaini, Y. S. L.-K. Cio, C. Leblond-Ménard, and M. Raison, "Fast scene analysis using vision and artificial intelligence for object prehension by an assistive robot," *Eng. Appl. Artif. Intell.*, vol. 63, pp. 33–44, Aug. 2017.

[14] Y.-S.-L.-K. Cio, M. Raison, C. L. Menard, and S. Achiche, "Proof of concept of an assistive robotic arm control using artificial stereovision and eye-tracking," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 12, pp. 2344–2352, Dec. 2019.

[15] H. Jiang, T. Zhang, J. P. Wachs, and B. S. Duerstock, "Enhanced control of a wheelchair-mounted robotic manipulator using 3-D vision and multimodal interaction," *Comput. Vis. Image Understand.*, vol. 149, pp. 21–31, Aug. 2016.

[16] H. Jiang, J. P. Wachs, and B. S. Duerstock, "Integrated vision-based robotic arm interface for operators with upper limb mobility impairments," in *Proc. IEEE 13th Int. Conf. Rehabil. Robot. (ICORR)*, Jun. 2013, Art. no. 6650447.

[17] H. Ka, D. Ding, and R. A. Cooper, "ARoMA-V2: Assistive robotic manipulation assistance with computer vision and voice recognition," in *Proc. 9th Conf. Rehabil. Eng. Assistive Technol. Soc. Korea*. Goyang, South Korea: RESKO, Nov. 2015. [Online]. Available: http://d-scholarship.pitt.edu/26361/

[18] C. P. Quintero, O. Ramirez, and M. Jägersand, "VIBI: Assistive vision-based interface for robot manipulation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 4458–4463.

[19] R. Atienza and A. Zelinsky, "Intuitive interface through active 3D gaze tracking," in *Proc. Int. Conf. Act. Media Technol. (AMT)*, May 2005, pp. 16–21.

[20] M. Buckley, R. Vaidyanathan, and W. Mayol-Cuevas, "Sensor suites for assistive arm prosthetics," in *Proc. 24th Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2011, pp. 1–6.

[21] M. Leroux, M. Raison, T. Adadja, and S. Achiche, "Combination of eyetracking and computer vision for robotics control," in *Proc. IEEE Int. Conf. Technol. Practical Robot Appl. (TePRA)*, May 2015, pp. 1–6.

[22] S. Li, X. Zhang, and J. D. Webb, "3-D-gaze-based robotic grasping through mimicking human visuomotor function for people with motion impairments," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 12, pp. 2824–2835, Dec. 2017.

[23] R. O. Maimon-Mor, J. Fernandez-Quesada, G. A. Zito, C. Konnaris, S. Dziemian, and A. A. Faisal, "Towards free 3D end-point control for robotic-assisted human reaching using binocular eye tracking," in *Proc. Int. Conf. Rehabil. Robot. (ICORR)*, Jul. 2017, pp. 1049–1054.

[24] D. P. McMullen et al., "Demonstration of a semi-autonomous hybrid brain–machine interface using human intracranial EEG, eye tracking, and computer vision to control a robotic upper limb prosthetic," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 784–796, Jul. 2014.

[25] C.-C. Postelnicu, F. Girbacia, G.-D. Voinea, and R. Boboc, "Towards hybrid multimodal brain computer interface for robotic arm command," in *Augmented Cognition* (Lecture Notes in Computer Science), vol. 1580. Cham, Switzerland: Springer, 2019, pp. 461–470, doi: 10.1007/978-3-030-22419-6_33.

[26] P. M. Tostado, W. W. Abbott, and A. A. Faisal, "3D gaze cursor: Continuous calibration and end-point grasp control of robotic actuators," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2016, pp. 3295–3300.

[27] E. Zahir, M. A. Hossen, M. A. A. Mamun, Y. M. Amin, and S. M. Ishfaq, "Implementation and performance comparison for two versions of eye tracking based robotic arm movement," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2017, pp. 203–208.

[28] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, pp. 16495–16519, 2017.

[29] Y. Cheng, H. Wang, Y. Bao, and F. Lu, "Appearance-based gaze estimation with deep learning: A review and benchmark," 2021, *arXiv:2104.12668*.

[30] E. Finnigan, "Eye gaze tracking for assistive devices," Ph.D. dissertation, Dept. EECS, Univ. California, Berkeley, CA, USA, May 2019.

[31] T. Fischer, H. J. Chang, and Y. Demiris, "RT-GENE: Real-time eye gaze estimation in natural environments," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 334–352.

[32] S. Park, A. Spurr, and O. Hilliges, "Deep pictorial gaze estimation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 721–738.

[33] S. Park, X. Zhang, A. Bulling, and O. Hilliges, "Learning to find eye region landmarks for remote gaze estimation in unconstrained settings," in *Proc. ACM Symp. Eye Tracking Res. Appl. (ETRA)*. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 1–10.

[34] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "It's written all over your face: Full-face appearance-based gaze estimation," May 2017, *arXiv:1611.08860*.

[35] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4511–4520.

[36] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Apr. 2014, *arXiv:1409.1556*.

[38] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Dec. 2001, pp. I-511–I-518.

[39] A. Adouani, W. M. Ben Henia, and Z. Lachiri, "Comparison of Haar-like, HOG and LBP approaches for face detection in video sequences," in *Proc. 16th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2019, pp. 266–271.

[40] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond neural face detection on mobile GPUs," Jul. 2019, *arXiv:1907.05047*.

[41] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," Apr. 2017, *arXiv:1704.04861*.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," Mar. 2019, *arXiv:1801.04381*.

[43] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[44] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile GPUs," Jul. 2019, *arXiv:1907.06724*.

[45] C. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," Jun. 2019, *arXiv:1906.08172*.

[46] K. Sokal. (Sep. 25, 2020). MediaPipe 3D Face Transform. Google LLC, Mountain View, CA, USA. [Online]. Available: https://developers.googleblog.com/2020/09/mediapipe-3d-face-transform.html

[47] X. Zhang, Y. Sugano, and A. Bulling, "Revisiting data normalization for appearance-based gaze estimation," in *Proc. ACM Symp. Eye Tracking Res. Appl.* New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 1–9.

[48] X. Zhang, Y. Sugano, and A. Bulling, "Evaluation of appearance-based methods and implications for gaze-based applications," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–13.

[49] A. Ablavatski, A. Vakunov, I. Grishchenko, K. Raveendran, and M. Zhdanovich, "Real-time pupil tracking from monocular video for digital puppetry," Jun. 2020, *arXiv:2006.11341*.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[51] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, p. 1–12.

[52] W. Falcon et al. (2019). *PyTorch Lightning*. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning

[53] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 120–123, 2000.

[54] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2015, *arXiv:1603.04467*.

[55] L. Wright and N. Demeure, "Ranger21: A synergistic deep learning optimizer," Aug. 2021, *arXiv:2106.13731*.

[56] L. Liu et al., "On the variance of the adaptive learning rate and beyond," Oct. 2019, *arXiv:1908.03265*.

[57] T. Fischer. (Jun. 2021). *Tobias-Fischer/RT_GENE*. [Online]. Available: https://github.com/Tobias-Fischer/rt_gene

[58] (Jul. 2021). *NVlabs/Few_Shot_Gaze*. [Online]. Available: https://github.com/NVlabs/few_shot_gaze

[59] N. Norori, Q. Hu, F. M. Aellen, F. D. Faraci, and A. Tzovara, "Addressing bias in big data and AI for health care: A call for open science," *Patterns*, vol. 2, no. 10, Oct. 2021, Art. no. 100347.