Eye Gaze Tracking

Nitin Kasshyap

Abstract—**This report presents a study on 3D eye gaze tracking using images. A neural network model is used to detect the gaze direction as a vector with the center of the eye as its origin. The trained model is then extended to estimate gaze from webcam input. A Perspective-n-Point solver is utilized to identify the 3D pose of the face with respect to the camera using an approximate face model. The computed gaze vector is also projected onto the computer screen to visualize the target point of the eye gaze.**

I. Introduction

Gaze is the point of vision and eye gaze tracking is the estimation of the direction a person's eyes are looking. In our everyday lives, we use eye gaze as a crucial non-verbal cue subconsciously, both to gain information from others as well as to signal our intentions. As a technology solution, the identification of gaze direction is a feature that can serve a variety of purposes. It can augment user input in the domain of human-computer interaction. Gaze also acts as the principal input for a variety of virtual- and augmented reality applications. Hence, the introduction of eye gaze tracking to real life systems could offer a mode of seamless interaction with intelligent systems.

Eye gaze estimation techniques have been investigated since the past few decades. The major focus of the input methods for the estimation systems were invasive at the beginning. Among the early non-invasive methods use visible [1] or infrared light sources [2] to solve this task. The purpose of the photocell was to measure the reflection of light off the eye caused by a fixed source. The difference in the trajectory of the reflected light from the incident ray was used to find the direction of gaze relative to the subject's head pose. The requirement of a dedicated light source and measurement in the above methods motivated the use of digital image-based techniques for capturing the eyes. The proposed technique in [3] uses image segmentation to identify the location of eyes, which are usually the darkest regions in a facial image. An iterative thresholding algorithm is used to estimate the locations of facial features and estimate gaze. Other work use conventional computer vision algorithms such as gradient of iris [4] as well as classification algorithms such as regression and support vector machines.

With the advent of machine learning, several attempts have been made to solve this task efficiently using advanced computer vision algorithms. The ability of high-dimensional extractors to learn to detect features in inputs with remarkable accuracy has led to the utilization of deep learning algorithms to solve applications in diverse domains. In [5], the authors use a convolutional neural network (CNN)-based model to build iTracker, a gaze detection system, and create GazeCapture, a large scale eye tracking dataset, for training such models. CNNs have also been used to detect eye movements and analyse them using the knowledge of 3D models of the eye and face [6]. The techniques that account for the approximate models of the eyes and the face are observed to be more accurate in learning features that represent gaze.

The goal of this project is to build a neural network-based eye gaze estimation model based on the work of [7]. The model is trained to predict the 3-D gaze direction in the eye coordinate frame. It is also designed ensuring the capability to be used for real-time gaze estimation without the need for any dedicated high performance computing resource after training. In order to gauge the proposed system's performance on extension to real-world applications, eye gaze detection on video captured by a web camera is studied. Reasonable performance of gaze estimation on the webcam video would indicate the feasibility of extension of the proposed approach to various real world applications.

II. Methodology

A. Neural Gaze Estimation

To predict the eye gaze from the normalized single eye image, a neural network model is trained. A convolutional network architecture is used due to its superior capabilities to extract features from an image while remaining spatially invariant. Two sets of problem setups are investigated, one with the eye image as its input and the other which also uses the corresponding head pose:

1) Head Pose-independent Gaze Estimation: The motivation to investigate the estimation of gaze without the head pose is to avoid the computational complexity involved in estimating the 3-D head pose from the 2-D input image. From a set of eye images $\boldsymbol{e_i}$, the goal of the training process is to learn a head pose independent function f that estimates gaze angle $\boldsymbol{a_p} = f(\boldsymbol{e_i})$ in the eye coordinate frame. The model is trained to minimise the mean squared error (MSE) between $\boldsymbol{a_p}$ and the set of target gaze angles $\boldsymbol{a_{t_i}}$ as

$$\min_f \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{a_{p_i}} - \boldsymbol{a_{t_i}})^2 \tag{1}$$

2) Gaze Estimation with Head Pose: From a set of eye images $\boldsymbol{e_i}$ and the corresponding head poses $\boldsymbol{h_i}$, the goal of the training process is to learn a function f that estimates gaze angles $\boldsymbol{a_p} = f(\boldsymbol{e_i}, \boldsymbol{h_i})$ in the eye coordinate frame. This model is also trained to minimise the MSE between $\boldsymbol{a_p}$ and the set of target gaze angles $\boldsymbol{a_{t_i}}$. The gaze direction is dependent on the pose of the eyes. The pupiliary distance introduces a parallax between the gaze directions of the two eyes. Hence, the model which learns to estimate gaze by knowing the head pose is expected to have better performance.

The architecture of the above two CNN models is based on the LeNet model [8], as suggested by the authors of the the MPIIGAZE dataset [7]. The use of a CNN with two convolutional and max-pooling layers each is justified in this situation due to the smaller size of the input eye image. Using a deeper model would require upscaling of the eye image, which could lead to a loss in the quality of features learned on training. Increased depth of the model would also cause the mapping function f to overfit on the training images e_i, possibly leading to the loss of generalization of the model and reduced performance during inference.

The estimated gaze angle $\boldsymbol{a_{p_i}}$ is then used to compute the estimated 3D unit gaze vector $\boldsymbol{g_{p_i}}$ in the eye coordinates, which is the desired output of the system [7]:

$$\begin{aligned} \boldsymbol{a_{p_i}} &= [\theta_i, \phi_i] && (2a)\\ y_i &= -sin(\theta_i) && (2b)\\ x_i &= -cos(\theta_i)sin(\phi_i) && (2c)\\ z_i &= -cos(\theta_i)cos(\phi_i) && (2d)\\ \boldsymbol{g_{p_i}} &= \frac{x_i\hat{i} + y_i\hat{j} + z_i\hat{k}}{\sqrt{x_i^2 + y_i^2 + z_i^2}} && (2e) \end{aligned}$$

The performance of the trained models are evaluated by using the metric of mean angle error, which is computed using the cosine similarity measure between the inner products of $\boldsymbol{g_{p_i}}$ and the unit target gaze vector $\boldsymbol{g_{t_i}}$:

$$error_{mean} = \frac{1}{m}\sum_{i=1}^{m} acos(\langle \boldsymbol{g_{p_i}} | \boldsymbol{g_{t_i}} \rangle) \quad (3)$$

The mean error angle is expected to serve as an indicator of the expected performance of gaze estimation on real world use.

B. Screen Gaze Estimation

The unit gaze vector $\boldsymbol{g_{p_i}}$ is computed with respect to the eye coordinate frame. To compute the on-screen location of gaze, the transformation between the eye coordinate frame and the screen coordinate frame is required. It is assumed that the screen plane is an $XY-$plane that is parallel to the $XY-$plane of the eye coordinate frame and displaced by a vector $\boldsymbol{z_s}$. Then, the gaze vector $\boldsymbol{g_{p_{iS}}}$ that intersects the screen plane is found in spherical coordinates $(r_{p_{iS}}, \theta_{iS}, \phi_{iS})$. The projection of this vector to the screen XY-plane, $\boldsymbol{g_{p_{iSXY}}}$, is the required screen gaze position.

$$\begin{aligned} \phi_{iS} &= acos\left(\frac{\langle \boldsymbol{g_{p_i}} | [0,0,1] \rangle}{\|\boldsymbol{g_{p_i}}\|}\right) && (4a)\\ \theta_{iS} &= atan2\left(\langle \boldsymbol{g_{p_i}} | [0,1,0] \rangle, \langle \boldsymbol{g_{p_i}} | [1,0,0] \rangle\right) && (4b)\\ r_{p_{iS}} &= \frac{|\boldsymbol{z_s}|}{cos(\phi_{iS})} && (4c)\\ g_{p_{iSXY}} &= r_{p_{iS}} sin(\phi_{iS}) && (4d)\\ x_{iSXY} &= g_{p_{iSXY}} cos(\theta_{iS}) && (4e)\\ y_{iSXY} &= g_{p_{iSXY}} sin(\theta_{iS}) && (4f) \end{aligned}$$

where $[x_{iSXY}, y_{iSXY}]$ is the desired gaze coordinate in the screen plane. When the screen coordinates are measured in a different system than the eye image, a homogeneous transformation operator T_{eSXY} can be used, $[x_{iS}, y_{iS}] = T_{eSXY}[x_{iSXY}, y_{iSXY}]^T$. As the proposed system assumes the use of a webcam with no additional information about the setup, finding T_{eSXY} algorithmically is not considered. The geometry of the above projection to the screen is show in Fig. 1.

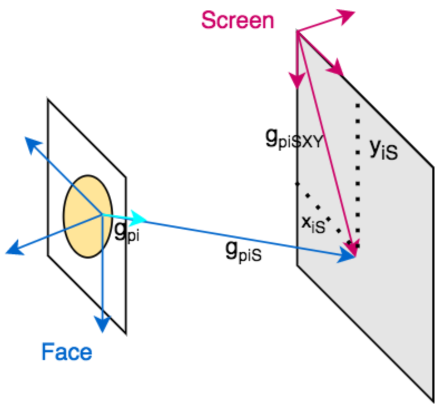

Fig. 1. Geometry of gaze projection to screen frame

C. Webcam Gaze Estimation

To use the neural gaze estimation model detailed in section II-A with images from a webcam, additional processing is essential. There are three coordinate frames involved in this task, as illustrated in Fig. 2.

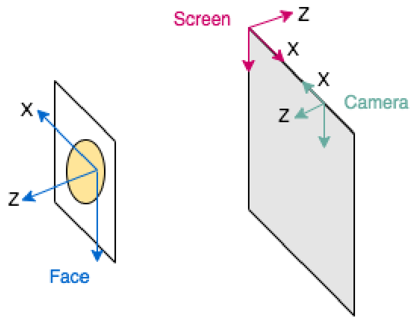

Fig. 2. The eye, webcam and the screen cooordinate frames.

Fig. 3 shows the pipeline designed for the task of transforming the webcam image to single eye image. The origin of the eye coordinate frame is defined at the center of the right eye, as per the convention followed by [9]. A face detection model is required to identify the face region in the webcam image. The face region is then used to compute the location of facial features including the corners of the eyes and mouth. A 6-point 3-D face model is used as an approximate representation of a general human face. Once the facial landmarks are identified

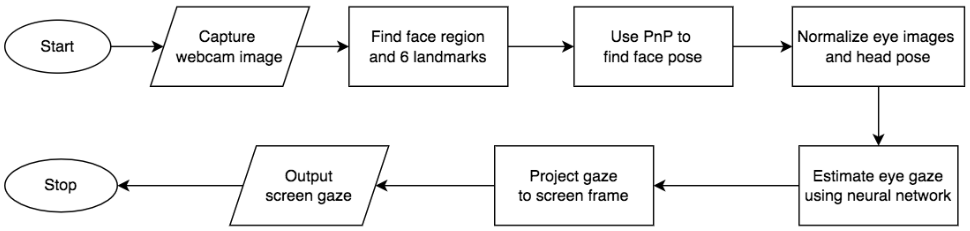
Start
Capture webcam image
Find face region and 6 landmarks
Use PnP to find face pose
Normalize eye images and head pose
Estimate eye gaze using neural network
Project gaze to screen frame
Output screen gaze
Stop

Fig. 3. Pipeline constructed to estimate screen gaze from webcam input.

[10] in the image, the Perspective-n-Point problem is solved to identify the approximate 6 DOF pose of the face in the camera frame. The Rodrigues operator is used to obtain the rotation vector corresponding to the rotation matrix and hence find the 3D pose of the head.

The face pose and transformation matrices are used to normalize the input webcam image. This step is essential to ensure the consistency of the webcam input with the images used to train the neural network model. Using the knowledge of camera parameters, a warp perspective transformation is used to obtain the cropped images of the eyes. These images and head poses can be used to obtain the estimate of gaze vector using the trained neural network. Once the gaze vector is found, it can then be transformed into the screen frame as detailed in Section II-B.

III. Results and Discussions

To implement the system, the Python programming language along with the image processing library OpenCV are used. Neural network models that accept eye images as inputs are created in the machine learning framework PyTorch. The MPIIGAZE dataset is used as the source of training inputs as well as test images. A wide range of inputs that include variations caused in real-world environments was a major reason for the choice of using this data. Another significant factor was the relatively smaller size of the corpus which made training on a personal computer feasible, unlike several recent ones that require dedicated high performance computing resources. As the MPIIGAZE dataset includes the normalized images and head pose vectors, they are directly fed into the training pipeline. The model is trained to minimise MSE using the Adam optimizer. Best practices such as early stopping and random weight initialization are also adapted to ensure optimal training performance. Once the models are trained, their performance is evaluated using mean angle error as defined in Section II-A. The obtained results after training are presented in Table I.

It can be observed from the table that the proposed neural network model performs well at the task of gaze estimation. The impact of training without head pose is seen as well, showing that the hypothesis of including head pose having a positive impact may not be necessary. Though the separation

TABLE I
Mean angle error on MPIIGAZE test samples.

No head pose	**With head pose input**	
	LeNet	**AlexNet**
$2.69°$	$2.81°$	$3.19°$

means that the left and the right eye have a different gaze vector for the same target, the impact of this separation is minimized due to the normalization process which includes perspective warping. However, a crucial factor to be noted here is that the head-pose independent model is not truly so —the 6-DOF pose was essential to compute and apply the warp perspective transformation.

To visualize the performance of the trained model, gaze estimates and the truth gaze directions are shown for inputs from the test set in Fig. 4. The projection of $\boldsymbol{g_{p_i}}$ onto the eye image plane $\boldsymbol{g_{p_{ieXY}}}$ is found as

$$x_{ieXY} = \langle \boldsymbol{g_{p_i}} | [1, 0, 0] \rangle \tag{5a}$$
$$y_{ieXY} = \langle \boldsymbol{g_{p_i}} | [0, 1, 0] \rangle \tag{5b}$$
$$\boldsymbol{g_{p_{ieXY}}} = [x_{ieXY}, y_{ieXY}] \tag{5c}$$

Then, $\boldsymbol{g_{p_{ieXY}}}$ is superposed on the eye image to help visualize the predicted gaze vectors. Here, the difference in the performance of models based on the head pose input can be observed. Fig. 5. shows the projected predicted eye gaze direction on some real-world test images by the pose-independent LeNet model. Though the metrics in Table I are not significantly different, the real-world performance of the gaze models are expected to be poor due to variations.

For webcam gaze estimation, the camera matrix and distortion coefficients are calculated. Dlib library then estimates the face bounding box and the relative locations of the six landmarks. The `solvePnP` algorithm in OpenCV is used to compute the face rotation matrix and the 6-D pose. Then, the face images are normalized as described in Section II-C to obtain the left and right eye images, which can be fed to the model to find the gaze vector.

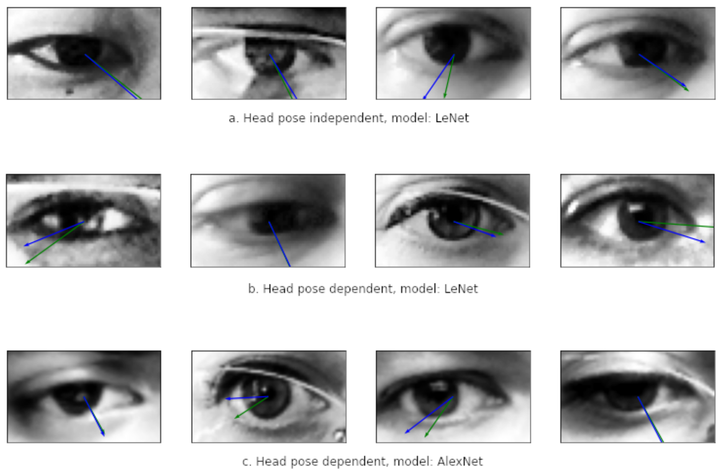
a. Head pose independent, model: LeNet
b. Head pose dependent, model: LeNet
c. Head pose dependent, model: AlexNet

Fig. 4. Predicted (blue) and true (green) gaze on MPIIGAZE test.

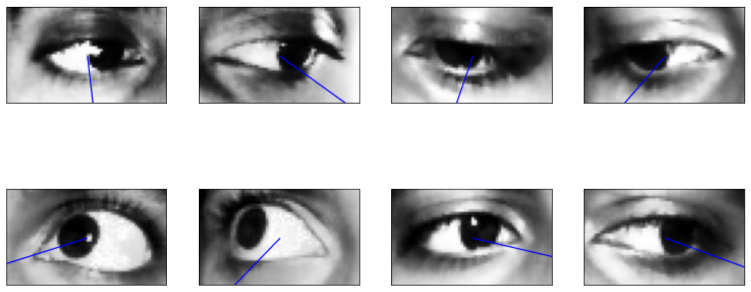

Fig. 5. Predicted (blue) gaze on real-world test.

IV. Conclusions

In this report, the study of a system to estimate the eye gaze from an image is presented. The system is designed to function without additional hardware or computation requirements. The trained neural network model succeeds at the task of gaze estimation with respect to the eye frame.

Future work on this project could focus on improving the performance of screen gaze. An improved normalization algorithm and head pose estimation using a 3-D face model with a greater number of feature points can be explored. Current work indicate that the use of temporal features in the inputs instead of treating them as a set of individual frames can lead to an improved and consistent performance of the system. With the availability of a suitable data corpus, this system can be applied towards tasks such as driver distraction and human emotion detection with minimal modifications.

References

[1] L. Stark, G. Vossius, and L. R. Young. "Predictive Control of Eye Tracking Movements". In: *IRE Transactions on Human Factors in Electronics* HFE-3.2 (Sept. 1962), pp. 52–57. DOI: 10.1109/thfe2.1962.4503342.

[2] Nicholas Torok, Victor Guillemin, and J. M. Barnothy. "Photoelectric Nystagmography". In: *Annals of Otology, Rhinology & Laryngology* 60.4 (Dec. 1951), pp. 917–926. DOI: 10.1177/000348945106000402.

[3] Rainer Stiefelhagen, Jie Yang, and Alex Waibel. "Tracking eyes and monitoring eye gaze". In: *Proc. Workshop on Perceptual User Interfaces*. 1997, pp. 98–100.

[4] R. Kothari and J.L. Mitchell. "Detection of eye locations in unconstrained visual images". In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 3. 1996, 519–522 vol.3. DOI: 10.1109/ICIP.1996.560546.

[5] Kyle Krafka et al. "Eye tracking for everyone". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2176–2184.

[6] Haoping Deng and Wangjiang Zhu. "Monocular Free-Head 3D Gaze Tracking with Deep Learning and Geometry Constraints". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3162–3171. DOI: 10.1109/ICCV.2017.341.

[7] Xucong Zhang et al. "Appearance-based Gaze Estimation in the Wild". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 4511–4520.

[8] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[9] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. "Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. USA: IEEE Computer Society, 2014, pp. 1821–1828. DOI: 10.1109/CVPR.2014.235.

[10] *Face Landmark Detection*. URL: http://dlib.net/face_landmark_detection.py.html.