Efficient monocular point-of-gaze estimation on multiple screens and 3D face tracking for driver behaviour analysis

Jon Goenetxea *, Luis Unzueta, Unai Elordi, Juan Diego Ortega, Oihana Otaegui

Intelligent Transport Systems and Engineering Department, Vicomtech, Paseo Mikeletegi 57, Donostia, Spain
*jgoenetxea@vicomtech.org

Abstract: In this work, we present an efficient monocular method to estimate the point of gaze (PoG) and the face in the 3D space of multi-screen driving simulator users, for driver behaviour analysis. It consists in a hybrid procedure that combines appearance and model-based computer vision techniques to extract the 3D geometric representations of the user's face and gaze directions. These are placed in the same virtual 3D space as those of the monocular camera and the screens. In this context, the intersections of the overall 3D gaze vector with the planes that contain each screen is calculated with an efficient line-plane intersection geometric procedure. Finally, a point-in-polygon strategy is applied to see if any of the calculated PoGs lies within any of the screens, and if not, the PoG on the same plane as that of the closest screen is provided. Experiments show that the error for the obtained PoG accuracy is reasonable for automotive applications, even in the uncalibrated case, compared to other state-of-the-art approaches, which require the user's calibration. Another advantage is that it can be integrated in devices with low computational capabilities, such as smartphones, with sufficient robustness for driver behaviour analysis.

1. Introduction

Typically, state-of-the-art eye gaze estimation techniques obtain the point of gaze (PoG) on one screen, only [1]. However, in the case of driving simulators there are usually more than one, e.g., one for the front view, one for each side view, another one for the dashboard, etc (Fig. 1). Besides, there can be different objects of interest at different locations of each screen and obtaining the gaze fixations and saccades, derived from the PoG, accurately on each screen is important for driver behaviour analysis [2]. Additionally, it is also preferable to simplify the installation and calibration of sensors and to reduce the power consumption as much as possible, avoiding alternative possibilities such as placing a dedicated PoG estimator for each screen. Thus, in our context, we only consider one monocular camera in front of the user and a humble CPU, e.g., those included in an embedded PC or a smartphone.

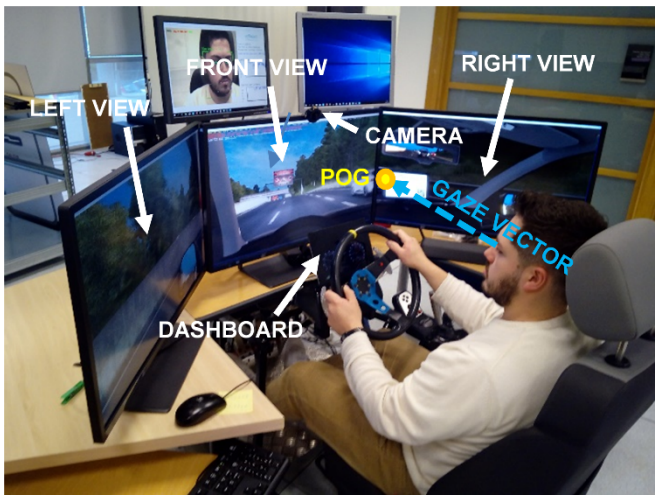

Fig. 1. *Multi-screen simulator setup for driver behaviour analysis, based on human-machine interaction, including PoG and 3D face tracking*

In automotive platforms, visual features of the face and eye regions of a driver provide cues about their degree of alertness, perception and vehicle control. Knowledge about driver cognitive state helps to predict, for example, if the driver intends to change lanes or is aware about obstacles and thereby avoid fatal accidents. These systems use eye tracking setups mounted on a car's dashboard along with computing hardware running machine vision algorithms, with computational capabilities far below from those of off-the-shelf desktop PCs. Major sources of error in automotive systems arise principally from platform and user head movements, variable illumination, and occlusion due to shadows or users wearing glasses, which need to be handled robustly but also efficiently due to the computational constraints.

The current state of the art of eye gaze estimation systems applied to automotive platforms includes different kind of approaches and uses. There are approaches that consider eye movement features (e.g., fixations, saccades, smooth pursuits, etc) for deriving driver cognitive states, such as driver distraction [3]. Other approaches apply classification techniques to eye images related with different gaze zones, to detect where the driver is looking at while driving [4]. There are also approaches that track facial features, 3D head poses and gaze directions relative to the car geometry to detect eyes-of-the road condition of the driver [5]. Other approaches study the driver's gaze behaviour (e.g., glance frequency and glance time) to evaluate the driving performance when they interact with other devices (e.g., a portable navigation system) while driving [6]. Finally, there are also approaches that study the dynamics between head pose and gaze behaviour of drivers to predict gaze locations from the position and orientation of a driver's head [7] or to categorise different kind of driver behaviours while driving [8].

Our main motivation in this work is to increase the grade of sophistication of all this kind of use cases by developing a more accurate, more robust, but still efficient

method for estimating the head pose and eye gaze of drivers, compared to previous approaches. We paid special attention to the case of multi-screen simulators, where the relation between the PoG and the rendered graphics can be directly established, and therefore, richer data could be extracted for behaviour analysis. In order to do so, it is necessary to relate the 2D image projections of the driver's facial and ocular cues, captured from the monocular camera, with the 3D space. Ideally, this would require not only obtaining the person's 3D eye gaze vectors from the images, but also the person's 3D eye positions and the surrounding potential targets' geometries in the same 3D space, the camera characteristics from which that space is observed, and an additional calibration stage done by the user. However, in many applications it is not easy to obtain all these data. This is the case of automotive applications, where it is not comfortable for the driver to spend time calibrating the eye gaze system. Other important factors are that the estimated gaze vector should have a low level of noise, but it should still be sensitive to quick eye movements, and that the estimated gaze vector should be robust to head movements, which in the case of driving, normally happen many times.

Our approach to tackle all these factors consists in a hybrid procedure that combines appearance and model-based computer vision techniques to extract the 3D geometric representations of the user's face and gaze directions. These are then placed in the same virtual 3D space as those of the monocular camera and the screens. This reconstructed virtual 3D world is where the driver's behaviour can then be analysed, based on the estimated PoG on the different targets of the scene and the 3D head pose, without necessarily requiring calibration data. It has been designed to have an acceptable balance between accuracy, robustness and efficiency, so that it can be integrated into devices with low computational capabilities that might be used in vehicles.

The rest of the paper is organised as follows. Section 2 introduces the proposed hybrid system. Section 3 illustrates details about our experiments and presents some discussions about them. Section 4 concludes the paper.

2. Methodology

The methods to estimate the eye gaze from monocular images and videos can be categorised in two types of approaches: model-based [5][9] and appearance based [4][8][10][11][12][13][14]. Next, we study more in detail the pros and cons of each and then we explain our proposed hybrid approach.

2.1. Model-based vs appearance-based

The model-based approach relies explicitly in 3D graphical models that represent the geometry of the eye (typically as spheres) which are fitted to the person's detected eye features in the image (typically, the iris and the eye corners). Thus, the fitted 3D model allows inferring the 3D eye gaze vector, which is then used to deduce where the person is looking at. These methods imply some drawbacks, such as: They require to precisely locate the iris of the eye in the image; this is often impossible, for example when the user's eyes are not wide open, which is the normal case. In order to estimate the eye gaze direction, they need the user's head coordinates system as reference. Therefore, the success of these methods is highly dependent on the precision with which the user's head coordinates system has been localised. Besides, although simple, they require an initialisation scheme: the user needs to intentionally look at one or more points on a screen. Otherwise, eye vectors cannot be obtained with sufficient precision. In sum, since they are pure geometric methods, their precision is strongly dependent on the precision of the estimated eyeball and pupil centres. However, common images do not enable to obtain this information with high precision.

On the contrary, the appearance-based approach establishes a direct relation between the person's eye appearance and the corresponding eye gaze data of interest (e.g., the 3D eye gaze vector) by applying machine learning techniques. Thus, a dataset of annotated images is used to train a regression model, which is then used to deduce where the person is looking at, when applied to the person's eye image extracted from the image.

In the last few years, the appearance-based methods have been greatly benefited by the revolutionary results obtained by the emerging deep learning techniques in computer vision applications and have become the current state of the art in the field. They allow to generalise much better the learned relation between the eye appearance and the corresponding eye gaze data than alternative machine learning approaches (based on "handcrafted" image features and "shallow" layered learning architectures), when a huge dataset of annotated images is used for training. Typically, hundreds of thousands or even millions of samples are used, which may include real data [10][14], photorealistic synthetic data [11][13] or even a mixture of both [12]. This way, eye gaze direction estimation systems can obtain better accuracies with people whose appearance has not been included in the training of the regression model.

However, an effective eye gaze direction estimation system does not only require obtaining accurate eye gaze data from eye images, but it also requires applying properly the eye gaze data to the environment, so that it is possible to deduce where the person is looking at.

2.2. Hybrid approach

Fig. 2 shows the general overview of the workflow of our approach, where the inputs are a monocular image grabbed by one camera in front of the user, a parametric deformable 3D face model (Fig. 3), the camera intrinsic parameters and the screen geometries. The outputs are his/her estimated PoG with respect to the considered screens and his/her facial mesh in the 3D space, which includes information about his/her head position, orientation and expression. In this workflow, we distinguish three blocks: (1) the 3D face model adjustment to the user's face image, (2) the normalisation of the 3D gaze estimation and (3) the estimation of the eye gaze direction with respect to the targets.

The first block comprises computer vision procedures to detect and track facial regions on the image, localise facial landmarks and fit the 3D face model to those landmarks, by optimising the following objective function:

$$e = \arg\min \frac{1}{n} \sum_{j=1}^{n} [d_j - p(f, w, h, \boldsymbol{t}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a})_j]^2 \qquad (1)$$

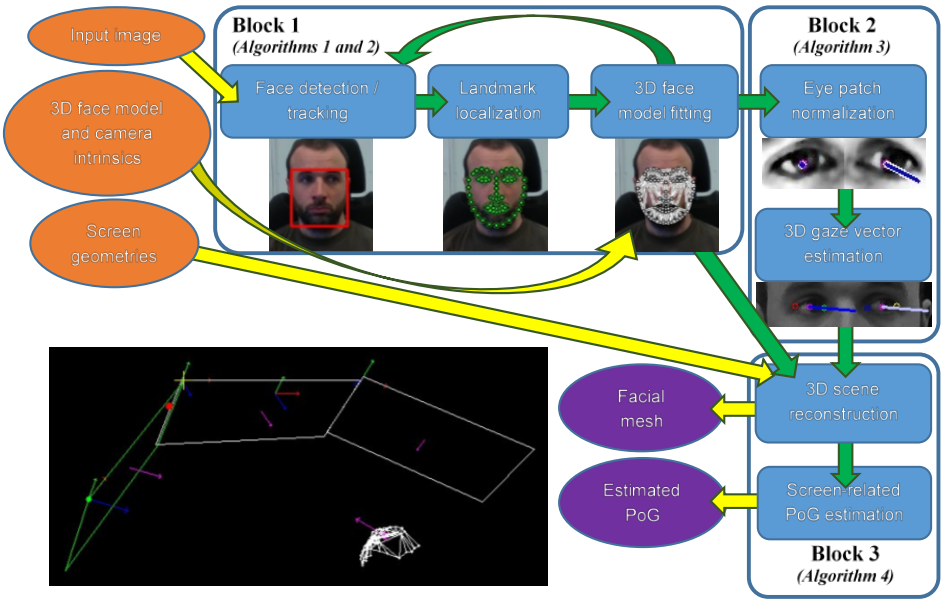

Fig. 2. *Workflow of the multi-planar PoG estimation and 3D face tracking approach*

where:

- $\boldsymbol{d} = \{d_1, d_2, d_3, \ldots\}$ are the detected 2D landmark positions.
- $\boldsymbol{p}= \{p_1, p_2, p_3, \ldots\}$ are the 2D projections of the corresponding 3D deformable model vertices. $\boldsymbol{p}$ is a function that depends on the camera parameters (f, w, h) and on the parameters of the graphical object ($\boldsymbol{t}$, $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{a}$). Function $\boldsymbol{p}$ represents the 2D projections on a surface of vertices, which are 3D. The goal is to minimise the distance between the detected 2D landmark positions in the image and the vertices of the projections.
- f is the focal length of the camera from which the image was obtained.
- w is the image pixel width.
- h is the image pixel height.
- $\boldsymbol{t}= \{t_x, t_y, t_z\}$ are the XYZ positions of the face model with respect to the camera.
- $\boldsymbol{r}= \{r_x, r_y, r_z\}$ are the roll-pitch-yaw rotation angles of the face model with respect to the camera.
- $\boldsymbol{s}= \{s_1, s_2, s_3, \ldots\}$ are the shape-related deformation parameters.
- $\boldsymbol{a}= \{a_1, a_2, a_3, \ldots\}$ are the action-related deformation parameters.
- n is the number of 2D landmark positions.
- e is the residual error.

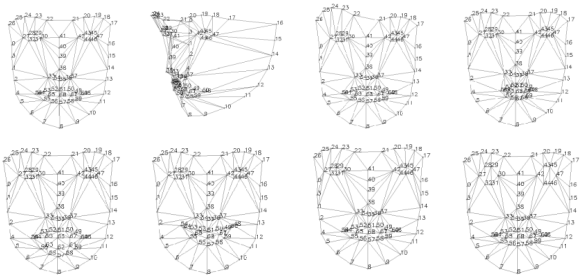

Fig. 3. *A generic deformable 3D face model and some of its deformation parameters compatible with our method*

For the localisation of the user's face region two stages are distinguished: (1) the initial face detection and posterior re-detections when the tracking is lost, and (2) the in-between face tracking. This is relevant as tracking algorithms typically are more efficient and require less memory than those for face detection. Thus, the face detection algorithm is only activated when the user's face is not being tracked. The detection is done with the SSD deep neural network [15], which has shown to be robust under challenging conditions, trained specifically with multiple-pose faces. The tracking is based on CLNF [16], applied at landmark level, which has a good balance between computational cost and localisation reliability and stability. The landmark distribution is constrained by a parametric 3D face model, to avoid impossible human facial shapes. The tracking is considered to be lost when the image under the face region does not correspond to a human face, according to the learned face pattern (see Algorithms 1 and 2 for further details).

Algorithm 1: Hybrid face model detection-tracking fitting algorithm

Input: ***The image sequence I***
Output: ***The face model parameters {t, r, s, a} that overlap the model to the user's face, throughout I***

For each $I_j \in \boldsymbol{I}$ **do**
  if *Face detection needed* **then**
    Reset the face model parameters of the graphical model to the neutral configuration
    Run the face region detector in the image
    Store the detected user's face image patch and face region
  else
    Locate a stored face image patch in the image (via pattern matching)
    Verify that the located patch corresponds to a real face (via pattern classification)
    if *Located face region contains a real face* **then**
      Store the located face region
    end
  end
  if *Face region available* **then**
    Run the face landmark detector in the face region
    Adjust the 3D face model to the detected landmarks ***(Algorithm 2)*** $\rightarrow \{\boldsymbol{t}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}\}_j$
  end
end
(Optional) Filter {***t, r, s, a***} with an appropriate approach for face movements

Algorithm 2: Three-stage face model adjustment algorithm

Input:
- ***Set of 2D landmark positions d in the image***
- ***The relation list between the landmark and vertices***
- ***The camera parameters {f, w, h}***

Output: ***The face model parameters {t, r, s, a} that overlap the model to the user's face***

1: Set the deformation parameters $\{\boldsymbol{s}, \boldsymbol{a}\}$ to zero
2: Convert the current parameter values to the normalised range workspace
3: Optimise, using for example the Levenberg-Marquardt algorithm [17][18], Eq. (1) with $\{\boldsymbol{t}, \boldsymbol{r}\}$ as the only variables
4: Optimise, using for example the BFGS algorithm [19][20][21][22], Eq. (1) with $\{\boldsymbol{s}\}$ as the only variables
5: **For each** $a_k \in \boldsymbol{a}$ **do**
6: Optimise, using for example the BFGS algorithm, Eq. (1) with $\{a_k\}$ as the only variable
7: **end**

Once the different facial parts are localised, the image regions around both eyes are extracted, and their shape and intensity distributions are normalised, so that a deep neural network, based on [10], can infer the corresponding 3D gaze vectors. Then, an overall gaze vector of the user is calculated as the weighted mean vector of both eyes with its origin at the midpoint of both eyes (see Algorithm 3).

Algorithm 3: Normalised left and right eye gaze vectors estimation algorithm

Input:
- ***The image sequence I***
- ***2D left $\{e_1, e_2\}_l$ and right $\{e_1, e_2\}_r$ eye corner landmark positions, throughout I***
- ***The adjusted face model geometry and parameters, throughout I***
- ***The pre-trained deep neural network for regressing 3D gazes from normalised eye images***

Output: ***The user's normalised left and right eye gaze vectors estimation $\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm}$, throughout I***

1: **For each** $I_j \in \boldsymbol{I}$ **do**
2: Calculate M for each eye (Eq. (2))
3: Obtain I_{norm}^{shape} for each eye (Eq. (3))
4: Obtain I_{norm} for each eye (via image equalisation)
5: Mirror I_{norm} for the eye not corresponding to that considered by the regressor (left or right)
6: Process both I_{norm} with the pre-trained deep neural network
7: Un-mirror the response for the mirrored eye image → $(\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm}^{reg})_j$
8: Apply the dominant eye and head rotation's correction factor (Eq. (4)) → $(\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm}^{corrected})_j$
9: Divide both regression results by their corresponding Euclidean norms → $(\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm})_j$
10: **end**

The affine transformation matrix M is calculated as follows:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot c_x - \beta \cdot c_y \\ -\beta & \alpha & \beta \cdot c_x + (1-\alpha)\cdot c_y \end{bmatrix} \quad (2)$$

where:

- $\alpha = s \cdot \cos(\theta)$
- $\beta = s \cdot \sin(\theta)$
- $s = (w - 2 \cdot m_1)$
- θ refers the horizontal rotation angle of the line that connects both eye corners.
- $\{c_x, c_y\}$ are the image coordinates of the centre of rotation in the source image.

Then, the source image I_{input} is transformed, that is to say, normalised in shape, using the matrix M, as follows.

$$I_{norm}^{shape}(x, y) = I_{input}(M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23}) \quad (3)$$

It must be noted that the applied eye shape normalisation procedure usually results in distorted images; normally, the further the user's face is with respect to frontal viewpoints, i.e., the most distant eye's appearance may look, normally, the more distorted the images become.

As a matter of example, Fig. 4 shows three examples of the distortion that happens in the normalised appearance of distant eyes in non-frontal faces, when the head's yaw angle is changed. As can be observed, the green points do not match exactly the white ones because the deformability of the graphical object is not perfect. At most, e is minimised (Eq. (1)). Consequently, this distortion may affect in stability of the estimated gaze for different yaw rotation angles of the head. A similar instability may also happen for different pitch angles, but in a lower degree.

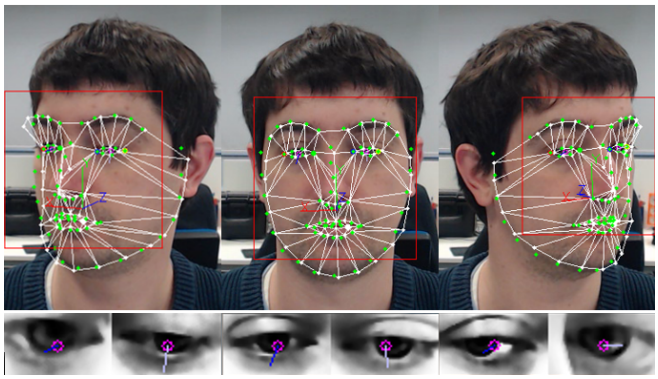

Fig. 4. *Examples of the distortion that happens in the normalised appearance of the most distant eyes in non-frontal faces, when the head's yaw angle is changed*

Thus, in order to reduce this effect, the vectors obtained in the previous step ($\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm}^{reg}$) are corrected by a factor that gives more importance to the dominant eye (the less distorted eye) and which is proportional to the head's pitch and yaw rotation angles, as follows:

$$\{\boldsymbol{g}_l, \boldsymbol{g}_r\}_{norm}^{corrected} = \{w_d \cdot \boldsymbol{g}_l, (1 - w_d) \cdot \boldsymbol{g}_r\}_{norm}^{reg} + \begin{Bmatrix} K_y \cdot (r_y - r_{y0}) \\ K_x \cdot (r_x - r_{x0}) \\ 0 \end{Bmatrix} \quad (4)$$

where:

- w_d is the weight of eye dominance.
- r_{x0} is the reference pitch angle.
- r_{y0} is the reference yaw angle.
- K_x is the proportionality constant for the pitch angle.
- K_y is the proportionality constant for the yaw angle.

In the case of big out-of-plane head rotations where both eye images are too distorted to be reliable, the gaze estimation relies solely on the head direction. The values of these parameters and ranges are experimentally determined, depending on the final application. For instance, the reference

pitch and yaw angles could be the average values from those observed during the image sequence, while the user's head poses are closer to frontal viewpoints, while the proportionality constants could be determined based on the observations of the gaze stability while the user is moving the head, but maintaining the point of gaze. Finally, each vector is divided by the Euclidean norm, so that to assure that the resulting vectors have unit norm, and this way both normalised gaze vectors are obtained.

It is remarkable that these 3D eye gaze vectors have been obtained without any previous calibration e.g. without any initialisation procedures. This is especially important in applications requiring real-time monitoring of the eye gaze, such as automotive applications.

Algorithm 4 shows how the eye gaze direction is estimated with respect to the targets. First, the target geometries are placed with respect to the camera's coordinate system, which is the same reference used for the face and eye gaze vectors, already estimated in previous blocks. The camera's coordinate system has been previously pre-established. In other words, it is assumed that the camera's coordinate system is well-known. A target is modelled or referred to as a set of polygons formed by k points $\boldsymbol{b}$ and lines $\boldsymbol{l}$, and their corresponding planar surfaces $\{\boldsymbol{v}, q\}$ (where $\boldsymbol{v}$ is the normal vector and q the distance from the origin) that define the objects that need to be related with the user's point of gaze (e.g., a screen is represented by a rectangular plane). Then, the 3D face model is placed in the scene with the obtained parameters. Then, the normalised left and right eye 3D gaze vectors are transformed, so that they are referred to the coordinate system of the camera (i.e., not to the normalised camera viewpoint, as before). This is done by removing the effect of the rotation angle θ that was used for the affine transformation applied to each normalised eye shape, like this:

$$\{g_l, g_r\} = \begin{Bmatrix} -\cos(\theta)\cdot(\{g_l,g_r\}_{norm})_x + sin\,(\theta)\cdot(\{g_l,g_r\}_{norm})_y \\ -\sin(\theta)\cdot(\{g_l,g_r\}_{norm})_x - cos\,(\theta)\cdot(\{g_l,g_r\}_{norm})_y \\ (\{g_l,g_r\}_{norm})_z \end{Bmatrix} \quad (5)$$

Then, both gaze vectors are combined by calculating its geometric mean g, which it is assumed to be the user's overall gaze vector. The gaze vector may optionally be filtered by taking into account its frame-to-frame motion and an appropriate filtering method for eye movements. The origin of this vector is preferably placed in the middle position (mean value) of both eye centres from the 3D face, Ɛ. Thus, the point of gaze PoG for each target plane can be estimated, like this:

$$\boldsymbol{PoG_t} = \text{Ɛ} + \frac{(\boldsymbol{q}-\boldsymbol{v}\cdot\text{Ɛ})}{\boldsymbol{v}\cdot\boldsymbol{g}}\cdot\boldsymbol{g} \quad (6)$$

Finally, a point-in-polygon strategy [23] is applied to see if any of the calculated PoGs lies within any of the screens. As can be observed, the point-in-polygon strategy may result in that the PoG goes through a polygon, or that it does not go through any polygon. If it does not go through a polygon, the method provides the closest polygon. For example, in line 11 of Alg. 4, if the PoG does not go through a polygon, the distance to the polygon is stored. And in line 12, the current measured distance is compared to the minimum measured distance (which is the stored one), in order to guarantee that the closest polygon is finally selected.

Algorithm 4: Target-related point of gaze estimation algorithm

Input:
- ***The set of polygons formed by k points b and lines l, plane normal vectors v and plane distances q with respect to the camera that represent the target objects*** $\{\boldsymbol{b_k}, \boldsymbol{l_k}, \{\boldsymbol{v}, \boldsymbol{q}\}\}_t$
- ***The adjusted face model geometry and parameters {t, r, s, a}, throughout I***
- ***The user's normalised left and right eye gaze vectors estimation*** $\{\boldsymbol{g_l}, \boldsymbol{g_r}\}_{norm}$***, throughout I***

Output: ***The user's PoG with respect to the targets in the scene, throughout I***

1: Place the target polygons with $\{\boldsymbol{b_k}, \boldsymbol{l_k}, \{\boldsymbol{v}, \boldsymbol{q}\}\}_t$
2: **For each** $I_j \in \boldsymbol{I}$ **do**
3: Place the 3D face model with $\{\boldsymbol{t}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}\}_j$
4: Transform $\{\boldsymbol{g_l}, \boldsymbol{g_r}\}_{norm}$ with Eq. (5) → $\{\boldsymbol{g_l}, \boldsymbol{g_r}\}_j$
5: Calculate the geometric mean vector → $\boldsymbol{g}_j$
6: (Optional) Filter $\boldsymbol{g}_j$ with an appropriate approach for gaze movements
7: $d_t^{min} = BIG\_NUMBER$
8: **For each** target ***t*** **do**
9: Calculate the point of gaze $\boldsymbol{PoG_t}$ in the target's plane (Eq. (6))
10: Apply a point-in-polygon strategy to the target's polygon
11: **If** point-in-polygon test successful ***then*** → $\boldsymbol{PoG_j}$ and **break**
12: **Else** store $\boldsymbol{PoG_t}$ and distance to polygon d_t
13: **If** $d_t < d_t^{min}$ ***then*** → $\boldsymbol{PoG_j} = \boldsymbol{PoG_t}$
14: $d_t^{min} = d_t \rightarrow \boldsymbol{PoG_j} = \boldsymbol{PoG_t}$
15: **end**
16: **end**
17: **end**

3. Results

We have evaluated our approach with an experiment where 8 people have been recorded by a camera in front of them, while using a driving simulator with three screens (Fig. 1). The participants were requested to look at different control points located at zones of interest on the screens: (1) left window, (2) left side mirror, (3) horizon, (4) road, (5) navigation panel, (6) rear mirror and (7) right side mirror (Fig. 5). They were free to rotate their head as they considered (no instructions were given about this). The accuracy of our approach has been measured in this setup without including a user-calibration stage. Thus, if the PoG obtained directly as explained above lies within the targeted zone of interest, it is

Fig. 5. *The considered zones of interest in the simulator to analyse the driver's PoG.*

Table 1 Comparison among different state-of-the-art eye gaze estimation systems and ours

Method category	Paper reference	Setup	Accuracy metrics (Mean % and/or º)
Model-based	[5]	1 camera, 1 IR illuminator and 18 gaze zones, considering day (no-glasses/glasses/sun-glasses) and night (no-glasses/glasses) scenarios	>95% on-the-road >90% off-the-road (for all scenarios)
	[9]	3 cameras (2 facing driver, 1 looking out) and 6 gaze zones	94.9%
Appearance-based	[4]	1 camera and 8 gaze zones	92.75%
	[8]	1 camera and 6 gaze zones	94.6%
	[14]	1 camera and 20 on-screen positions	10.8º (cross-dataset evaluation)
Hybrid	Ours	1 camera and 7 gaze zones	97.0% / 4.6º (front screen) 87.7% / 11.5º (side screens)

considered a correct response, wrong otherwise. Besides, we measured the angle between the vector that goes from the head to the targeted control point and from the head to the estimated PoG.

Table 1 shows the obtained results, along with those obtained by other state-of-the-art model-based [5][9] and appearance-based [4][8][14] alternatives with similar setups and conditions. Ideally, we would have reimplemented and adapted to our setup all these approaches so that then we could measure the differences under the same working conditions. However, taking into account that there are many implementation details that are not available in the publications, which can be important for the reproduction of the reported results, we have preferred to include them here directly with their corresponding setups and accuracy metrics. In some cases, they are given in degrees between the estimated and ground-truth gaze vectors and in other with a percentage of the number of times in which the correct gaze zones are reached. In our case, we provide both metrics so that it is easier to compare with the other approaches, despite the differences among the setups and conditions. Nevertheless, note that due to that reason this comparison is more qualitative than quantitative, except for their own setups with respect to their corresponding ground truth measurements.

[9] has the most different setup as it uses two cameras to capture the driver's data, and hence, it has the possibility of estimating 3D features directly and thus improve the accuracy, compared to the monocular case. However, we prefer to avoid this kind of setups in order to simplify the installation and configuration (i.e., calibration) and reduce the power consumption.

In the case of [5], it follows a similar scope to ours, using facial feature tracking, 3D head pose and gaze estimation, but with some relevant differences. The head pose estimation algorithm is based on the 'weak-perspective' assumption, which with the kind of images obtained in this setup produces an inherent error due the orthographic projection that needs to be compensated. On the other hand, its proposed gaze vector estimation procedure is model-based, which has the drawbacks already stated before.

Both [4] and [8] rely on classifiers trained with the relations between gaze zones and feature descriptors composed by 2D facial part and ocular image cues. The drawback of this kind of approaches is that, as they do not estimate 3D data, they need to be specifically trained for each setup, and provide more limited information for behaviour analysis.

[14] relies on a deep neural network to estimate the 3D gaze vector, in similar way as we do, but including both the normalized eye appearance and the head orientation as input data for the network. In this case, the approach is evaluated with people looking at a laptop screen, so no profile views are contemplated like those that occur in our case when users look at the side screens and the eye appearances get distorted.

In our case, it can be seen that we obtain sufficient accuracy to relate rendered graphics with the user's observations, despite not having calibrated the system for each user. As expected, the accuracy is lower for the side screens, but still high enough (Fig. 6). Anyway, these errors should be considered when designing the recognition areas for the interaction with the elements of the scene, i.e., for higher errors the area of interaction around the element should be bigger too. Fig. 7 shows that our approach can handle quick eye movements, but maintaining a low level of noise for fixations.

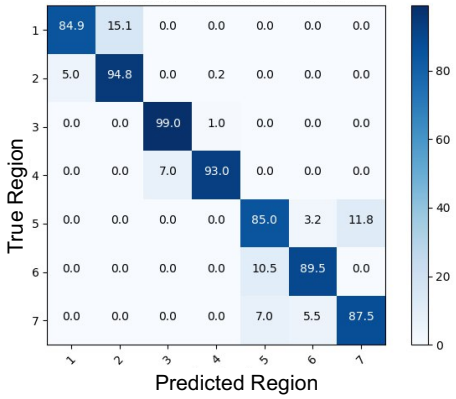

Fig. 6. *Confusion matrix of the predictions obtained by our approach for the considered gaze zones*

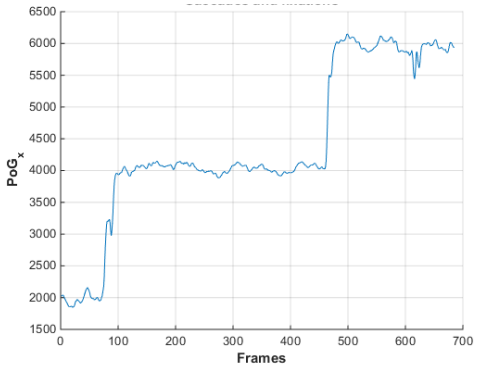

Fig. 7. *An example of PoG$_x$ signal where saccades and fixations can be appreciated, along with the level of noise*

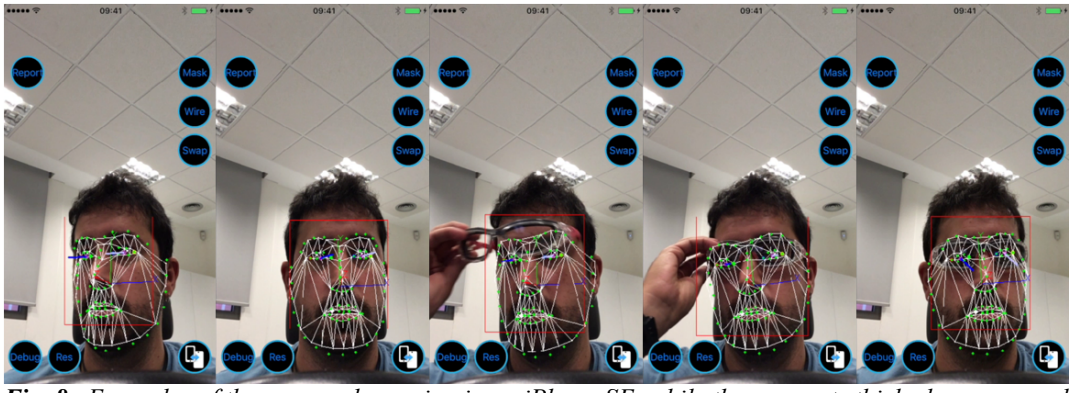

Fig. 8. *Examples of the approach running in an iPhone SE, while the user puts thick glasses on and the system keeps working*

On the other hand, in order to evaluate the efficiency of our approach and its suitability for running it in devices which can then be used in real vehicles, where one cannot expect installing CPUs/GPUs like those of desktop/laptop PCs, we have integrated it in an app for smartphones with iOS and Android operating systems (Fig. 8). It is remarkable to state that the operating system can also have an impact in the overall performance of the app, due to the multi-level structure and different programming languages in which the app needs to be programmed (i.e., the core of the approach is programmed in C++ for both operating systems, while the interface is in Objective-C for iOS and Java for Android). More specifically, we have tested the iOS app in an iPhone SE (with iOS 10.3.2) and the Android app in a Docomo smartphone (with Android 6). The measured average FPS (frames-per-second) of our app in each case has been 30 and 20 respectively, which reveals the efficiency and suitability of our approach to be applied in a real-world scenario.

4. Conclusion

One of the advantages of our approach is that with a simple setup we can efficiently estimate the PoG of the user in multiple screens of a simulator, allowing to relate directly the rendered graphics that represent the different elements of the scene with the user's observations.

Moreover, as the rendered scene simulates a physical car environment with a distribution close to a real case, this approach is suitable to be used inside a driving situation. In a real scenario, the zones in the rendered scene fit with the key attention zones considered while driving (with some variations depending on the car). This way it is easier to generate richer data for developing driving behaviour analysis approaches.

Another advantage is that it can be integrated, processed and executed in devices with low computational capabilities, such as smartphones.

Future work will principally focus on optimizing the deep neural network designs for the face detection, landmark localization and eye gaze vector estimation stages to further improve their efficiency in ARM-based CPUs.

We also plan to adapt the approach to real vehicle setups. The primary challenge in a real driving scenario is the illumination variability. Some image capture setups reduce the illumination issues using specific hardware like infrared cameras or a sort of optical filters. These changes call for particular datasets to re-train some of the models (e.g. eye gaze estimation model and face detection model), but the method pipeline is not affected.

5. Acknowledgments

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 690772, VI-DAS project).

6. References

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

[2] Kasneci, E., Kasneci, G., Kübler, T. C., et al.: 'Online recognition of fixations, saccades, and smooth pursuits for automated analysis of traffic hazard perception', Artificial Neural Networks, Springer Series in Bio-/Neuroinformatics, 4, 2015, pp. 411–434

[3] Liang, Y., Reyes, M. L., Lee, J. D.: 'Real-time detection of driver cognitive distraction using support vector machines', IEEE Trans. Intell. Transp. Syst., 8, (2), 2007, pp. 340–350

[4] Chuang, M.-C., Bala, R., Bernal, E., et al.: 'Estimating gaze direction of vehicle drivers using a smartphone camera', Proc. IEEE Conf. Comput. Vis. Pattern Recogn. Work. (CVPRW), 2014, pp. 165–170

[5] Vicente, F., Huang, Z., Xiong, X., et al.: 'Driver gaze tracking and eyes off the road detection system', IEEE Trans. Intell. Transp. Syst., 16, (4), 2015, pp. 2014–2027

[6] Zheng, R., Nakano, K., Ishiko, H., et al.: 'Eye-gaze tracking analysis of driver behavior while interacting with navigation systems in an urban area', IEEE Trans. Human-Mach. Syst., 46, (4), 2016, pp. 546–556

[7] Jha, S., Busso, C.: 'Analyzing the relationship between head pose and gaze to model driver visual attention', Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC), 2016, pp. 1–6

[8] Fridman, L., Lee, J., Reimer, B., et al.: 'Owl and Lizard: Patterns of head pose and eye pose in driver gaze classification', IET Comput. Vis., 10, (4), 2016, pp. 308–313

[9] Tawari, A., Chen, K. H. and Trivedi, M. M.: 'Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation', Proc. Int. IEEE Conf. Intell. Transp. Syst., 2014, pp. 988–94

[10] Zhang, X., Sugano, Y., Fritz, M. et al.: 'Appearance-based gaze estimation in the wild', Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR), 2015, pp. 4511–4520

[11] Wood, E., Baltrusaitis, T., Zhang, X., et al.: 'Rendering of eyes for eye-shape registration and gaze estimation', Proc. IEEE Int. Conf. Comp. Vis. Pattern Recogn. (CVPR), 2015, pp. 3756–3764

[12] Shrivastava, A., Pfister, T., Tuzel, O., et al.: 'Learning from simulated and unsupervised images through adversarial training', Proc. IEEE Conf. Comp. Vis. Pattern Recogn. (CVPR), 3, 2017, pp. 2242–2251.

[13] Ranjan, R., De Mello, S. and Kautz, J.: 'Light-weight head pose invariant gaze tracking', Proc. IEEE Conf. Comp. Vis. Pattern Recogn. (CVPR), 2018, pp. 1–9

[14] Zhang, X., Sugano, Y., Fritz, M., et al.: 'MPIIGaze: Real-world dataset and deep appearance-based gaze estimation', IEEE Trans. Pattern Anal. Mach. Intell., 2017

[15] Liu, W., Anguelov, D., Erhan, D., et al.: 'SSD: Single shot multibox detector', Proc. European Conf. Comput. Vis. (ECCV), 2016, pp. 21–37

[16] Baltrusaitis, T. Morency, L.-P., Robinson, P.: 'Constrained local neural fields for robust facial landmark detection in the wild', IEEE Conf. Comput. Vis. Workshops (ICCVW), 2013, pp. 354–361

[17] Levenberg, K.: 'A method for the solution of certain non-linear problems in least squares', Quarterly of Appl. Math., 2, 1944, pp. 164–168

[18] Marquardt, D.: 'An algorithm for least-squares estimation of nonlinear parameters'. SIAM J. Appl. Math., 11, (2), 1963, pp. 431–441

[19] Broyden, C. G.: 'The convergence of a class of double rank minimization algorithms: 2. The New algorithm', J. Inst. Math. Appl., 6, 1970, pp. 222–231

[20] Fletcher, R.: 'A new approach to variable metric algorithms', Computer J., 13, 1970, pp. 317–322

[21] Goldfarb, D.: 'A family of variable metric methods derived by variational means', Math. Comp., 24, 1970, pp. 23–26

[22] Shanno, D. F.: 'Conditioning of quasi-Newton methods for function minimization', Math. Comp., 24, 1970, pp. 647–650

[23] Haines, E.: 'Point in polygon strategies', in Heckbert, P. (Ed.): 'Graphics Gems IV' (Academic Press, 1994), pp. 24–46