Single image face orientation and gaze detection

Article *in* Machine Vision and Applications · November 2009

DOI: 10.1007/s00138-008-0143-1 · Source: DBLP

3 authors, including:

Jeremy-Yirmeyahu Kaminski
Holon Institute of Technology
28 PUBLICATIONS **259** CITATIONS

Machine Vision and Applications (2009) 21:85–98
DOI 10.1007/s00138-008-0143-1

ORIGINAL PAPER

Single image face orientation and gaze detection

Jeremy Yrmeyahu Kaminski · Dotan Knaan · Adi Shavit

Received: 23 September 2007 / Accepted: 1 April 2008 / Published online: 13 June 2008

Abstract We introduce a system to compute both head orientation and gaze detection from a single image. The system uses a camera with fixed parameters and requires no user calibration. Our approach to head orientation is based on a geometrical model of the human face, and is derived form morphological and physiological data. Eye gaze detection is based on a geometrical model of the human eye. Two new algorithms are introduced that require either two or three feature points to be extracted from each image. Our algorithms are robust and run in real-time on a typical PC, which makes our system useful for a large variety of needs, from driver attention monitoring to machine-human interaction.

Keywords Intelligent driver support · Human activity recognition · Visual tracking · Target detection · Vision system

1 Introduction

The computation of head orientation and gaze direction is a key component in applications such as car driver attention monitoring and human–computer interface for medical or multimedia purposes.

J. Y. Kaminski (✉)
Department of Computer Science, Holon Institute of Technology, Golomb Street, Holon, Israel
e-mail: kaminsj@hit.ac.il

D. Knaan · A. Shavit
EyeTech Corporation, Israel Labs, Jerusalem, Israel
e-mail: dotan@eyetech.jp

A. Shavit
e-mail: adish@eyetech.jp

Such systems run, by definition, in a complex environment due to variations in human faces and behaviors, variations in scene illumination and other dynamic environmental factors.

We propose simple and efficient algorithms for computing both head orientation and gaze direction. Our algorithm successfully handles these challenges while using a simple setting consisting of a single, fixed-focus camera and requiring no user calibration.

Based on our algorithm for computing head orientation, we present a method of computing gaze. We describe an overall system which performs gaze detection from a single image. Furthermore, in extreme situations, where the angle between the subject's face and the camera axis is too large to allow detection of all the needed facial features, we introduce a second algorithm for computing the gaze from a single eye.

We present experimental results showing that our system is both accurate and fast, which makes it suitable for a large variety of applications.

1.1 Comparison with other approaches

Before introducing our method, we present a short comparison with previous works. The problem of head position and gaze detection has received a huge amount of attention in the past decade, and we do not pretend to establish a comprehensive review of previous work. However, we consider hereafter what seems to be the most relevant references to show the novelty of our approach.

The large number of proposed algorithms for gaze detection indicates that no solution is completely satisfying. [5], is a good survey of several gaze detection techniques. In [9,12], a stereo system for gaze and face pose computation is presented, which is particularly suitable for monitoring driver vigilance. Both systems are based on two cameras, a narrow field camera (which provides a high-resolution image of

the eyes by tracking a small area) and a large field camera (which tracks the whole face). Besides the computationally complex difficulties arising from the use and control of multiple cameras, the system hardware is quite costly. In [11], a monocular system is presented, which uses a personal calibration process for each user and does not allow large head motions. Head motion restriction is typical for systems that utilize a single camera. Ohno et al. [11] use a auto-focus (motorized) lens to estimate the distance of the face from the camera. In [15], the eyegaze is computed using the fact that the iris contour, while being a circle in 3D is an ellipse in the image due to perspective projection. The drawback in this approach is that a high-resolution image of the iris area is required. This severely limits the allowed motions of the user, unless an additional wide-angle camera is used.

In this paper we introduce a new approach with several advantages. The system is monocular, hence the difficulties associated with multiple cameras are avoided. The camera parameters are maintained constant in time. The system requires no personal calibration and the head is allowed to move freely. This is achieved by using a model of the face and eye, deduced from anthropometric features. This kind of method has already received some attention in past [3,4,7]. However, our approach is simpler, requires less points to be tracked and is, eventually, more robust and practical.

In [3,4], the head orientation is estimated under the assumption of the weak perspective image model. This algorithm works using four points: the mouth corners and the external corners of the eyes. Once these points are precisely detected, the head orientation is computed by using the ratio of the lengths L_e and L_f, where L_e is the distance between the external eye corners and L_f the distance between the mouth and the eyes. In [7], a five-point algorithm is proposed to recover the 3D orientation of the head, under full perspective projection. The internal and external eye corners provide four points, while the fifth point is the bottom of the nose. The first four points approximately lie on a line. Therefore the authors use the cross-ratio of these points as an algebraic constraint on the 3D orientation of the head. It is worth noting that using cross-ratio requires a lot of caution. A good estimate of the cross ratio can only be obtained when the minimum distance of the adjacent pairs of the four collinear points is sufficiently large with respect to the known level of image noise [8].

In contrast, our approach to head orientation is based on three points only and works with a full perspective model. The three points are the eye centers and the middle point between the nostrils. Using these three points, we can compute several algebraic constraints on the 3D head orientation, based on a anthropomorphic model of the human face. These constraints are explicitly formulated in Sect. 2. Once the head orientation is recovered, further computations are possible. In this paper, we show an application to gaze detection. Furthermore, in situations where facial features are only partially detectable, we introduce an efficient and accurate method for gaze computation from a single eye. Some of these results appeared in a previous and shorter paper [10].

Our approach of a mechanically simple, automatic and non-intrusive system, allows eye-gazing to be used in a variety of applications where eye-gaze detection was not an option before. For example, such a system may be installed in mass-produced cars. With the growing concern of car accidents, customers and regulators are demanding safer cars. Active sensors that may prevent accidents are actively pursued. A non-intrusive, cheaply produced, one-size-fits-all eye-gazing system could monitor driver vigilance at all times. Drowsiness and inattention can immediately generate alarms. In conjunction with other active sensors, such as radar, obstacle detection, etc., the driver may be warned of an unnoticed hazard outside the car.

Psycho-physical and psychological tests and experiments with uncooperative subjects such as children and/or primates, may also benefit from such a static (no moving parts) system, which allows the subject to focus solely on the task at hand while remaining oblivious to the eye-gaze system.

In conjunction with additional higher-level systems, a covert eye-gazing system may be useful in security applications. For example, monitoring the eye-gaze of ATM clients. In automated airport check in counters, such a system may alert of suspiciously behaving individuals.

The paper is organized as follows. In Sect. 2, we present the face model that we use and how this model leads to the computation of the Euclidean face 3D orientation and position. We present simulations, which show the results are robust to error in both the model and the measurements. Section 3 introduces the overall system, combining head orientation computation and gaze detection. Section 3.4 focuses on a two-eye algorithm for gaze detection, while Sect. 3.5 presents an alternative algorithm allowing the use of a single eye for gaze detection. This alternative may be used either on its own or in interaction with the whole system. An integrated system using both approaches is introduced in Sect. 3.6. Finally, Sect. 4 presents some experiments.

2 Face model and geometric analysis

2.1 Face model

Following statistical data taken from [1], we assume the following model of a generic human face. Let **A** and **B** be the centers of the eyes, and let **C** be the middle point between the nostrils. Then we assume the following model:

$$d(\mathbf{A}, \mathbf{C}) = d(\mathbf{B}, \mathbf{C}) \tag{1}$$

$$d(\mathbf{A}, \mathbf{B}) = \rho d(\mathbf{A}, \mathbf{C}) \tag{2}$$

$$d(\mathbf{A}, \mathbf{B}) = 6.5 \text{ cm} \tag{3}$$

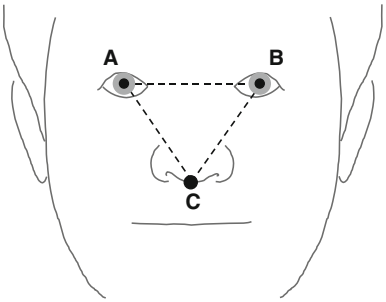

Fig. 1 The face model is essentially based on the fact that the two eyes and the nose bottom form an *isosceles triangle*

where $\rho = 1.0833$ and

$$d(\mathbf{M}, \mathbf{N}) = \sqrt{(x_M - x_N)^2 + (y_M - y_N)^2 + (z_M - z_N)^2}$$

is the Euclidean distance between the two 3D points **M** and **N**. The two first equations allow computing the orientation of the face, while the third equation is necessary for computing the distance between the camera and the face. The face model is illustrated in Fig. 1. The data gathered in [1] shows that our model is widely valid over the human population. Of course variations do exist, but the simulations presented in Sect. 2.4 show that our algorithm is quite robust over the whole spectrum of human faces.

2.2 3D face location

Let **M** be the camera matrix. All the computations are done in the coordinate system of the camera. Therefore the camera matrix has the following expression:

$$\mathbf{M} = \mathbf{K}[\mathbf{I}; \mathbf{0}],$$

where **K** is the matrix of internal parameters (see, e.g., [2,6]). Note that we did not consider the model with radial distortions. As detailed in the sequel, both simulations and experiments made from real images show that no significant loss of precision is generated using the above classical pinhole model.

Let (**a**, **b**, **c**) be the projection of (**A**, **B**, **C**) onto the image. In the equations below, the image points **a**, **b**, **c** are given by their projective coordinates in the image plane, while the 3D points **A**, **B**, **C** are given by their Euclidean coordinates in $\mathbb{R}^3$. Given these notations, the projection equations are:

$$\mathbf{a} \backsimeq \mathbf{KA} \tag{4}$$
$$\mathbf{b} \backsimeq \mathbf{KB} \tag{5}$$
$$\mathbf{c} \backsimeq \mathbf{KC} \tag{6}$$

where $\backsimeq$ means equality up to a scale factor. Therefore the 3D points are given by the following expressions:

$$\mathbf{A} = \alpha \mathbf{K}^{-1}\mathbf{a} \tag{7}$$
$$\mathbf{B} = \beta \mathbf{K}^{-1}\mathbf{b} \tag{8}$$
$$\mathbf{C} = \gamma \mathbf{K}^{-1}\mathbf{c} \tag{9}$$

where α, β, γ are unknown scale factors.[1]

Plugging these expressions of **A**, **B** and **C** into Eqs. (1) and (2), leads to two homogeneous quadratic equations in α, β, γ:

$$f(\alpha, \beta, \gamma) = (\gamma \mathbf{K}^{-1}\mathbf{c} - \alpha \mathbf{K}^{-1}\mathbf{a})^2 - (\gamma \mathbf{K}^{-1}\mathbf{c} - \beta \mathbf{K}^{-1}\mathbf{b})^2 = 0 \tag{10}$$

$$g(\alpha, \beta, \gamma) = (\beta \mathbf{K}^{-1}\mathbf{b} - \alpha \mathbf{K}^{-1}\mathbf{a})^2 - \rho(\gamma \mathbf{K}^{-1}\mathbf{c} - \alpha \mathbf{K}^{-1}\mathbf{a})^2 = 0 \tag{11}$$

Note that we used the following shortcut. For a column vector v, we wrote v^2 for $v^t.v$. Thus finding the points **A**, **B** and **C** is now essentially finding the intersection of two conics in the projective plane, since a conic is defined by a quadratic equation. Moreover, since no solution is on the line defined by $\gamma = 0$ (since the nose of the user is not located at the camera center!), one can reduce the computation of the affine piece defined by $\gamma = 1$. Hence we shall now focus our attention on the following system:

$$f(\alpha, \beta, 1) = 0 \tag{12}$$
$$g(\alpha, \beta, 1) = 0 \tag{13}$$

This system defines the intersection of two conics in the affine plane. We present in Appendix a way to compute the intersection points of these two conics. The computation is done in two stages. First we compute the *resultant* with respect to y, then given the resultant's roots, we compute the solutions of the system. All the concepts and the technical backgrounds are given in Appendix.

2.3 3D face orientation

We solve the system S defined by Eqs. (12) and (13) using the approach presented in Appendix. By Bezout's theorem [14]

[1] These could also be deduced by considering the points at infinity of the optical rays generated by the image points **a**, **b**, **c** and the camera center. These points at infinity are simply given in projective coordinates by: $[\mathbf{K}^{-1}\mathbf{a}, 0]^t$, $[\mathbf{K}^{-1}\mathbf{b}, 0]^t$, $[\mathbf{K}^{-1}\mathbf{c}, 0]^t$. Then the points **A**, **B**, **C** are given in projective coordinates by $[\alpha\mathbf{K}^{-1}\mathbf{a}, 1]^t$, $[\beta\mathbf{K}^{-1}\mathbf{b}, 1]^t$, $[\gamma\mathbf{K}^{-1}\mathbf{c}, 1]^t$. These expressions naturally yield the Eqs. (7)–(9) giving the Euclidean coordinates of the points.

(or simply by looking at the degree of the resultant), we know that there are at most four complex solutions to this system. Experiments show that a system generated by the image of a human face has only two real roots. In our setting, the ambiguity between these two roots is easily handled, since one solution leads to non realistic eye-to-eye distance. Furthermore, in case the ambiguity does not allow a clear choice, one can select the correct solution using the knowledge of the previous solution. This can be embedded into a simple tracking procedure in the three-dimensional space. In our system, both cues are used and there is no situation where the system makes a wrong choice. Let (α_0, β_0) be the correct solution. Then the points $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are known up to a unique scale factor. We shall denote $\mathbf{A}_0, \mathbf{B}_0$ and $\mathbf{C}_0$ the points obtained by the solution (α_0, β_0), Thus we have the following expression:

$$\mathbf{A}_0 = \alpha_0 \mathbf{K}^{-1}\mathbf{a} \tag{14}$$

$$\mathbf{B}_0 = \beta_0 \mathbf{K}^{-1}\mathbf{b} \tag{15}$$

$$\mathbf{C}_0 = \mathbf{K}^{-1}\mathbf{c} \tag{16}$$

Thus we have the following relations too: $\mathbf{A} = \gamma \mathbf{A}_0$, $\mathbf{B} = \gamma \mathbf{B}_0$ and $\mathbf{C} = \gamma \mathbf{C}_0$.

The computation of γ is done using the third model equation (3). Once the face points are computed, we can compute the distance between the user's face and the camera and so the 3D orientation of the face. Indeed the normal to the plane defined by $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ is given by:

$$\overrightarrow{N} = \overrightarrow{AB} \wedge \overrightarrow{AC},$$

where $\wedge$ is the cross product.

2.4 Robustness to errors in model and detection

We performed several simulations in order to estimate the sensitivity of this algorithm to errors in model and in detection. We simulated a high-resolution camera, in accordance to our initial setting, as explained in Sect. 3.1. Therefore in the simulation, we consider:

- The focal length $f = 4{,}000$ in pixels (which is very close to the value of the high-resolution camera used initially in the system),
- The principal point is at the image center,
- The distance between the camera and the face is 60 cm (which is a realistic setting of the system).

The simulations are done according to the following protocol. An artificial face, defined by three points in space, say $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, is projected onto a calibrated camera. Given a parameter p, we perform a perturbation of p by a white Gaussian noise of standard deviation σ. For each value of σ, we perform 100 random perturbations. For each value of p, obtained by this process, we compute the error in the 3D reconstruction as the mean of the square errors.

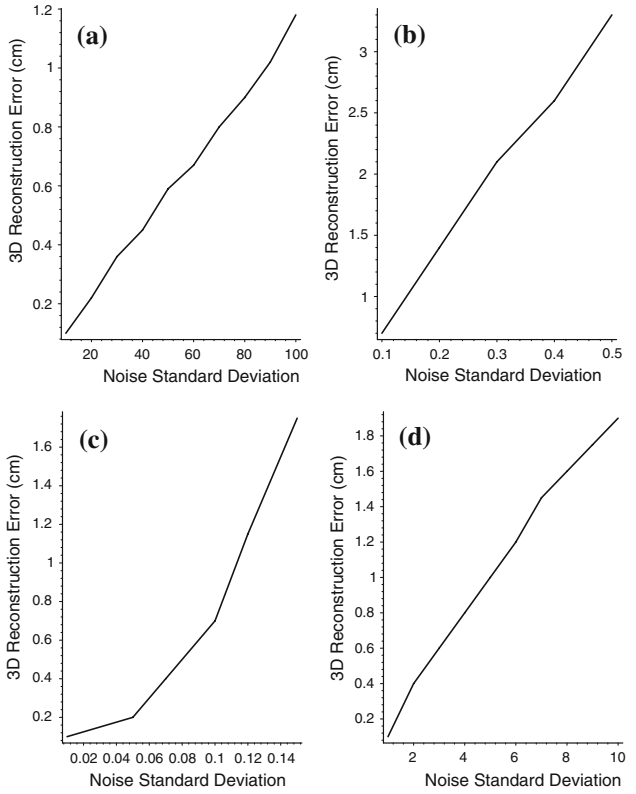

Fig. 2 The influence of different parameters on the 3D reconstruction. **a** Influence of the error in focal length. **b** Influence of the error in inter-eyes distance. **c** Influence of the error in human ratio. **d** Influence of the error in image points

The first simulation (see Fig. 2a) shows that the system is very robust to errors in the estimation of the focal length. Indeed, for a noise with standard deviation of 100 (pixels), which amounts to be an error of 2.5% in the focal length, the reconstruction error is 1.2 cm, less than 2% of the distance between the camera and the user.

The next two simulations aim at measuring the influence of errors in the model. First, the assumed eye-to-eye distance is corrupted by a Gaussian white noise (Fig. 2b). The mean value is 6.5 cm as mentioned in Sect. 2. For a standard deviation of 0.5, which represents an extreme anomaly with respect to the standard human morphology, close to a 8% deformation (see [1]), the reconstruction error is about 3.3 cm, which is close to 5% of the distance between the camera and the user. The influence of the human ratio r, as defined in Eq. (2), is also tested by adding a Gaussian white noise, centered at the mean value 1.0833 (Fig. 2c). For a standard deviation 0.15, which also represents a very strong anomaly (close to 14%), the reconstruction error is 1.75 cm,

less than 3% of the distance between the camera and the user.

After measuring the influence of errors in camera calibration and model, the next step is to evaluate the sensitivity to input data perturbation. The image points are corrupted by a Gaussian white noise (Fig. 2d). In our context, the typical distance between the two eyes in the image is between 300 and 400 pixels. For a noise of 10 pixels, which is a quite large error in detection (about 2.85% of the actual location), the reconstruction error is less than 2 cm, about 3% of the distance between the camera and the user.

The accuracy of the system is mainly due to the fact that the focal length is high ($f = 4{,}000$ in pixels). Indeed, when computing the optical rays generated by the image points, as in Eqs. (7)–(9), we use the inverse of K, which is roughly equivalent to multiplying the image points coordinates by $1/f$. Hence the larger f is, the less impact a detection error has on the computation.

3 Gaze detection

In this section, we show how the ideas presented above can be used to build a gaze detection system that does not require any user calibration or interaction.

3.1 System architecture

The main practical goal of this work was to create a non-intrusive gaze detection system, that would require no user cooperation while keeping the system complexity low. We initially used a high-resolution 30 fps, $1{,}392 \times 1{,}040$ video camera with a 25-mm fixed-focus lens. The CCD pixel is a square of length equal to 6.45 μm. Thus the focal length is 3,875, which is the same order of magnitude as the value used in the simulation. This setup allows both a wide field of view for a broad range of head positions, and high-resolution images of the eyes. Since we can estimate the 3D head position from a single image, we can use a fixed focus lens instead of a motorized auto-focus lens required by others [11] for distance estimation. A fixed focus lens is a cheaper and makes the camera calibration simpler, the calibration of the internal parameters being done only once. In Sect. 4.1, we will relax the requirement of a high-resolution camera and show that similar results can be obtained with a standard VGA resolution camera.

The system also uses IR LEDs at known positions to illuminate the user's face. For the two-eyegaze detection algorithms, one IR LED is enough, while for the single-eyegaze algorithm two IR LEDs are required. These different configurations are described below in more detail.

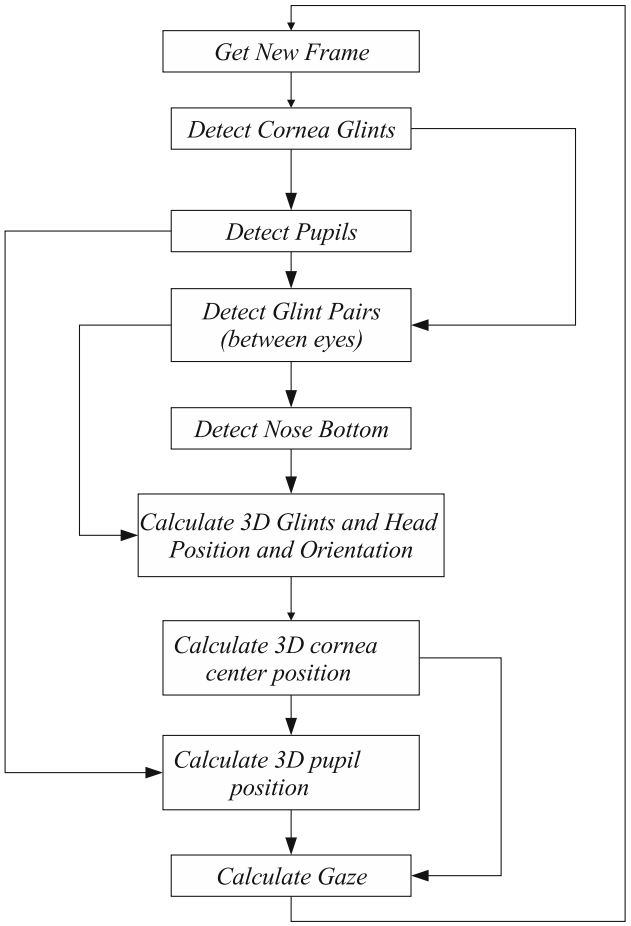

Fig. 3 The system flow chart, showing the different stages of the process

3.2 System overview

The general flow of the system is depicted in Fig. 3. This chart presents the first variant of the system for which gaze is detected using two eyes. The second variant, which mixes one and two-eye algorithms for gaze detection is introduced in Sects. 3.5 and 3.6.

For every new frame, the *glints*, which are the reflections of the LED light from the eye corneas as seen by the camera, are detected and their corresponding pupils are found. The search area for the nose is then defined, and the nose bottom is found. Given the two glints and nose position, we can reconstruct the complete Euclidean 3D face position and orientation relative to the camera, using the geometric algorithm presented in Sect. 2. This reconstruction gives us the exact 3D position of the glints. Then, for each eye, the 3D cornea center is computed using the knowledge of the LED position, as shown in Fig. 6. This model is similar to the eye model used in [11]. We then proceed to the 3D pupil detection. This, combined with the cornea center, finally provides the gaze direction.

The following subsections will describe these stages in more detail.

3.3 Feature detection

3.3.1 Glint and pupil detection

In our system, the distance between the camera and the face is about 1m, so that the face covers a large part of the image. Thus there is no need to perform face detection. One can directly proceed to the detection of features within the face. If face detection is required, there exists plenty of works in the literature to perform this. For an integrated approach to face detection and facial features localization, one can see [16].

In our system, the detection of the glints is done in several steps. Glints appear as very bright dots in the image, usually at the highest possible gray-scale values. We based our detection on these two properties: small area (a few pixels) and very high brightness. Moreover geometric constraints are incorporated.

Thus the first stage of the selection is to detect thin edges, or more precisely small objects, bright in respect to their immediate background. The detection works separately along the x and y directions, then the algorithm fusions the results. For each direction (x or y), we proceed in two steps. First we detect what we call "rising" edges, from dark areas to white and bright areas. Then we detect "falling" edges from dark to bright areas. Then we perform on both edge images a morphological dilatation. Finally we keep only the areas of overlapping between the two dilated edge images. Whence the procedure is applied to both axes, we merge the results by keeping only the overlapping. This results is a very robust precise and robust detection of thin edge. The final binary image is used a mask where glint candidates appear as white pixels.

Once this mask is computed, we filter the candidates according to the following properties: (i) brightness: only very bright candidates are kept (usually about 10% of all the candidates), (ii) size: only small candidates are kept (The concept of small depends on the camera resolution. In our setting, this is about 15 pixels.), (iii) areas crowded with candidates are filtered out (they are suspected as hair), (iv) false glints on eyeglasses are filtered out because they have bright neighboring pixels.

Whence the candidates have been filtered by the abovementioned procedure, there are still several possible locations for each glint. Then we incorporate into the selection geometric constraints which allow to select the right candidate. Since we want to detect a glint per eye, we are looking for two glints. Using the information from the previous frame, and under the very common assumption that the frame rate is quite faster than the head motion, we can have an excepted value for the coordinates of the each glint. This is made even more robust using a Kalman filter. Therefore we pair glints and keep the pairs that are likely right according to the output of the tracking. For each of these happy few candidates, we try to select pupils and irises as explained below. Finally only one pair yields acceptable pupils and irises. We select this one.

The detection of pupils or irises is done as follows. The pupils serve two purposes. They are used to filter out incorrect glint pairs, as mentioned previously, and they are required for the calculation of the gaze direction in the later stages of the algorithm. Pupils appear as round or oval dark regions inside the eye and are very close to (or behind) the glints. We consider a rectangle surrounding the glint and we look for an optimal gray value threshold that produces within this rectangle the roundest dark blob. In order to find this optimal threshold, we sample an interval of likely thresholds and for each value, we proceed to the following steps: (i) we create a binary image of the rectangle, (ii) we compute blobs in this binary image, (iii) we filter out blobs that are not round enough or that are too far from the glint, too small or too large. The final blob is compared with final blobs resulting from other threshold value. We select the blob that is the most round, the most dark, the closest to the glint and whose size is similar than the one selected in the previous frame. Finally, the pupil is an ellipse which approximates the convex hull of the best blob.

Once we have the pupil ellipse, we look for the iris around it. We expect it to be a similar ellipse but at a different scale. More precisely the ellipse the iris defines is centered at the same point that the pupil, has the same aspect ratio (the ratio between the major axis and the minor axis), but is bigger. We select a range of scale factors which are likely equal the ratio between the iris axes and the pupil axes. For each scale factor in this given range, we compute the median gray level of the pixels within the ellipse. Finally we select the scale whose difference from the next scale is the maximal. Heuristically, this mean that we found the ellipse that surrounds the most closely the iris. The next ellipse would include significant portion of the sclera (white background in the eye).

It is worth noting that the using IR LEDs also allows treating the case where the subject wears sunglasses. Indeed there three types of sunglasses: (i) those that are completely transparent to IR light, (ii) semi-transparent sunglasses,and (iii) blocking sunglasses.

The first kind of sunglasses looks in IR images as regular eyeglasses. There is no reason to handle this kind of sunglasses in a special way. As for the second kind, they block IR light partially. In this case, the eyes are not detected when the image is captured with regular exposure time, but can be seen when the exposure time is extended. An example is given in Fig. 17. The last kind is more problematic. Indeed blocking sunglasses block IR light completely. The eyes cannot be seen through these sunglasses. In this case, we need to compromise: a simple approach is described below.

Only the second type requires a special treatment. This special algorithm is considered only when the eyegaze is

not detected. In this case we proceed to the following steps: (i) we sum the rows of the image and project them on the vertical axis, (ii) we sum the columns of the image and project them on the horizontal axis. These projections provide a histogram-like analysis in the x and y directions. We schematically represent these operations in Fig. 4. If the subject wears sunglasses, we expect to get a bright-dark-bright pattern around the eyeglasses on the vertical axis and dark-bright-dark pattern around the eyeglasses on the horizontal axis. These patterns define image areas which are likely to contain sunglasses. We effectively check that these areas are covered by dark pixels.

If sunglasses are detected, then we extend the exposure time until eyegaze is detected. (This solution of course will work for sunglasses of the second type.) If eyegaze is not detected even with maximal exposure time, then the sunglasses are of the third type. In this case our software is set into blocking mode, and does not try to detect eyegaze anymore until the sunglasses are removed. The blocking mode is important in order to prevent false eye detection. In such a case, we can use the face orientation as eyegaze approximation: the glass centers are considered as the centers of the eyes, and together with the nose, we compute the face orientation as described in Sect. 2.

3.3.2 Nose detection

The detection of the nose-bottom is done by searching for dark-bright-dark patterns in the area just below the eyes. Indeed, the nostrils appear as dark blobs in the image thanks to the relative position of the camera and the face as shown in Fig. 5. The size and orientation of this search area is determined by the distance and orientation of the chosen glint-pair. Once dark-bright-dark patterns are found, we use connected component blob analysis on this region to identify only those dark blobs that obey certain size, shape, distance and relative angle constraints that yield plausible nostrils. The nose bottom is selected as the point just between the two nostrils.

3.4 Gaze detection from two eyes

Given the glints and the bottom point of the nose, one can apply the geometric algorithm presented in Sect. 2 to compute the 3D face orientation. As seen in Sect. 2.4, even if the glints are not exactly located in the center of the eye, the system returns an accurate answer. Then, for each eye, the cornea center is computed using the knowledge of the LED position, as shown in Fig. 6. By the Descartes-Snell law, if the cornea is modeled as a sphere, the cornea center lies on the bisector of the angle defined by the LED, the glint point in 3D and the camera. Its exact location is given by the cornea radius, which is 0.77 cm.

Fig. 4 The sunglasses detection procedure is conceptually represented here. *Rows* are projected onto the vertical axis, while *columns* are projected onto the horizontal axis

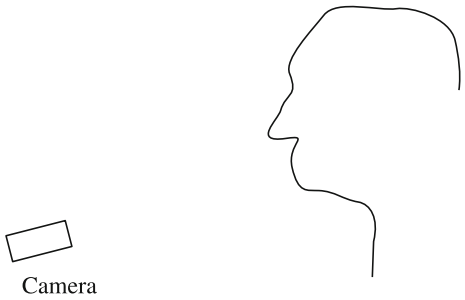

Fig. 5 The camera is viewing the eyes and the nostrils

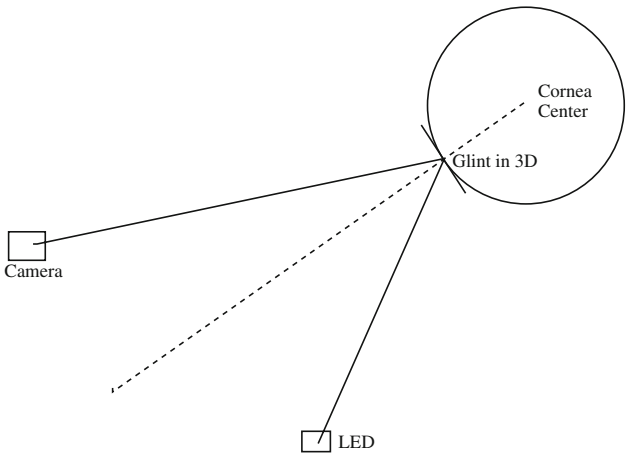

Fig. 6 The cornea center lies on the bisector of the angle defined by the LED, the glint point in 3D and the camera. Its exact location is given by the cornea radius, which is 0.77 cm

The gaze line is defined as being the line joining the cornea center and the pupil center in 3D. See Fig. 7 for a graphic view of this.

The pupil center is first detected in the image and then computed in 3D. The detection of the 2D pupil center is done by delimiting the whole pupil. For this task, a seed is detected as a dark point with dark neighbors in the eye. Then from this seed, we use a region-growing algorithm in all directions which is stopped when the homogeneity of the region is corrupted by the iris area of the eye. The result is the border of the pupil, which is an almost circular curve. The center of this circle is the 2D pupil center. Next we compute

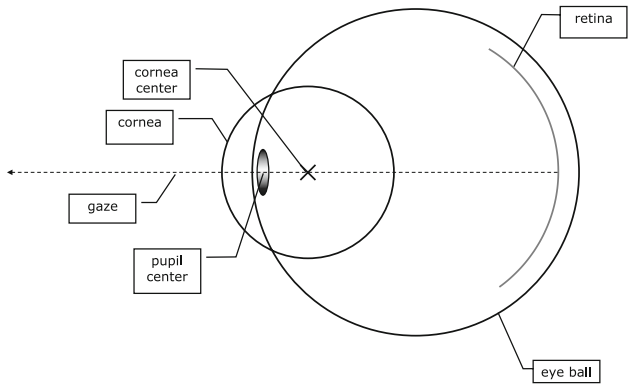

Fig. 7 An idealized view of the cornea. The *gaze line* is the line joining he cornea center and the pupil center in 3D

the spatial location of the of the pupil. Here again, we use morphological data. The distance between the pupil and the cornea center is a known measure of the human anatomy, 0.45 cm. Consider then a sphere S centered at the cornea center, with a radius of 0.45 cm. The pupil center lies on the optical ray generated by its projection onto the image and the camera center. This ray intersects the sphere S at two points. The closest of these points to the camera is the pupil center in space. The gaze line can then be extracted easily.

3.5 Gaze detection from a single eye

In this section, we show how one can achieve an accurate gaze detection from a single eye. This algorithm is important in situations where only one eye of the subject is detected. This occurs when the head rotation with respect to the camera is large, or in case of occlusions. This algorithm can be used on its own or can be used integrated into the whole system. Our system uses a combination of the two approaches. This will be further discussed in Sect. 3.6.

We shall now describe how gaze detection from a single eye is possible when two LEDs are used. Indeed, in order to perform this computation, we need to compensate for the a priori lack of information by another source of knowledge. This is done by using a second LED. Since two LEDs are used, two glints appear on each eye. It turns out that in practice, these two glints appear very close to each other.

Two steps are necessary to recover the gaze in this context. First we compute the cornea center. Second, the gaze direction is recovered in a similar way as in Sect. 3.4.

3.5.1 Setting

As mentioned above, the second method uses a different configuration. Two LEDs are used, L_1 and L_2, which produce two glints on each eye. Let us consider only one eye and the two image glints, g_1 and g_2, produced on this eye. Let O be the camera center, which is also the origin of the world coordinate system. As in Sect. 2.2 , let K be the internal parameter matrix of the camera.

The cornea center of this eye must lie on the line, defined as the intersection of the two following planes: $\overline{OL_1g_1}$ and $\overline{OL_2g_2}$. Refer to Fig. 6 to see that the cornea center indeed belongs to each of these planes.

3.5.2 Computing the direction of the cornea center

The following 3D vector $V_1 = K^{-1}g_1$ defines the direction of the optical ray generated by the glint g_1 (where g_1 is in homogeneous coordinates). Hence any $3D$ point P projected onto g_1 in the image has Euclidean coordinates given by λV_1, where λ is some scalar.

Similarly $V_2 = K^{-1}g_2$ defines the direction of the optical ray generated by g_2.

The cornea center C must satisfy the following equation (where the coordinates are all Euclidean):

$$C = \alpha L_1 + \beta V_1 = \gamma L_2 + \delta V_2$$

This equation expresses the fact that C lies on both planes $\overline{OL_1g_1}$ and $\overline{OL_2g_2}$. Since O is at the origin, being on the plane $\overline{OL_1g_1}$ is equivalent to be a linear combination of L_1 and V_1. The same remarks holds for $\overline{OL_2g_2}$. This finally yields the above equation.

Therefore the following must hold:

$$\begin{aligned}&\det([\alpha L_1 + \beta V_1, L_2, V_2])\\ &\quad = \alpha\det([L_1, L_2, V_2]) + \beta\det([V_1, L_2, V_2])\\ &\quad = 0\end{aligned}$$

Therefore, choose:

$$\begin{cases}\alpha = -\det(V_1, L_2, V_2)\\ \beta = \ \det(L_1, L_2, V_2)\end{cases}$$

Then the point $C_0 = \alpha L_1 + \beta V_1$ is some point on the optical ray of the cornea center. For the sake of simplicity, we normalize C_0, such that $\|C_0\| = 1$. Hence there exists some $\lambda \in \mathbb{R}\backslash\{0\}$, such that $C = \lambda C_0$.

3.5.3 Computing the cornea center

We must now compute λ. This is done by an iterative process similar to binary search. We first make sure that C_0 is located in front of the camera. Otherwise we multiply it by -1. Then we consider the right-side glint on the eye. Then starting with $\lambda = 1$, we proceed to the following algorithm:

1. Compute the cornea center: $C(\lambda)$.
2. Initialize parameters $\theta = -1$ and $\delta = 10$.
3. Compute the glint candidate in space. For the sake of simplicity, let us denote it $G(\lambda)$. This is done by intersecting

the optical ray generated by the glint on the right side of the eye with the sphere surrounding the cornea center, of radius equal to the cornea radius. We consider the intersection point which is the closest to the camera center.

4. Compute the bisector of the angle defined by the camera center O, the glint candidate in space $G(\lambda)$ and the corresponding LED. Let us denote it **b**.
5. Compute the normal to the cornea at $G(\lambda)$: **n**.
6. Both **b** and **n** are oriented toward the interior of the angle. Compute the angle θ from **n** to **b**.
   (a) If θ has the same sign as in the previous iteration, then $\lambda \leftarrow \lambda - \text{sign}(\theta)\delta$.
   (b) If the sign of θ has changed with respect to the previous iteration, then $\delta \leftarrow \delta/2$ and $\lambda \leftarrow \lambda - \text{sign}(\theta)\delta$.
7. Repeat from step 3, until the absolute value of θ is smaller than a predefined threshold.

This algorithm is based on the fact that the angle θ should be zero at the right glint candidate. At each iteration, the computed center is moved by a step of $\pm\delta$ in order to minimize $|\theta|$.

Once the cornea center is computed, the algorithm proceeds as described in Sect. 3.4 to compute the gaze.

3.5.4 Robustness analysis

As in Sect. 2.4, we investigate here single eye,and two LED detection robustness. In this context, since two LEDs are used, as we already mentioned in Sect. 3.5.1, two glints are generated over a single eye. Below, we shall call these glints a pair of *twin glints*. Then given these twin glints on the eye we tested how errors in different parameters influence the 3D detection of the cornea center. We tested the effect of errors in the following parameters:

1. The 2D position of the detected glints in the image.
2. The camera focal length.
3. The 3D LEDs locations.

For each parameter, we used the same protocol as in Sect. 2.4. We randomly generated 100 cornea centers normally distributed around (0, 0, 60)—the typical cornea center. For each cornea center, we computed the 2D twin image glints, based on a virtual camera and two virtual LED positions. Given the computed glints, the different factors are corrupted by Gaussian noise, with zero mean. Several values of the standard deviation were tested with 100 samples each. We then computed the median error induced in the reconstruction of the 3D cornea center for these samples. Note that the maximal standard deviation presents very extreme cases. For example, the radius of a glint in a typical image is three pixels. Therefore an error of three pixels is very rare. Subpixel errors are much more typical.

The robustness of the algorithm is clearly reflected in Fig. 8a–c. The dot in each graph marks the typical standard deviation seen in our system.

A similar experiment was conducted to test the effect of the same parameters on the computed eye-gaze angle. The average error caused by the above parameters perturbations was less than 2° for all the parameters.

3.6 Integrated system

This subsection is devoted to show how both approaches (either one LED—two eyes or two LEDs—single eye) are used together.

The updated flow chart 9 shows how this collaboration is achieved. More precisely, the integrated system works as follows. When a new frame is captured, we first look for cornea glints and pupils around them. We then continue in two separate methods that we described:

1. If two eyes (both glint and pupil) were detected, we look for the nose, and then calculate the head orientation and the 3D position of the glints and cornea center in each eye.
2. In each eye we find twin glints (glints of the 2 IR LED sources), and use them to compute the 3D cornea centers of each eye separately.

For each eye, based on the two 3D cornea centers that were computed by the two methods, we computed an optimal cornea center, which results from the fusion of the two previous results. We found that gaze detection computation using the two LEDs approach with a single eye is even more robust than the single LED approach with two eyes. Therefore in the cooperation process, we give a higher reliance score to this approach. We then use the cornea center to compute the 3D pupil, and the eyegaze, as was described in Fig. 7.

4 Experiments

4.1 Confidence evaluation

In this section, we present a protocol that we used for an overall evaluation of the system. For this purpose, we built a measure of the orientation of a direction d in $\mathbb{R}^3$, not parallel to the plane $z = 0$. Let u_x, u_y, u_z be the unitary vectors along the axes x, y, z. We first project the direction d onto the plane $z = 0$: $d' = d - (d.u_z)u_z$. Then the angle $\alpha_x = \arccos(d'.u_x/\|d'\|)$ (respectively $\alpha_y = \arccos(d'.u_y/\|d'\|)$)

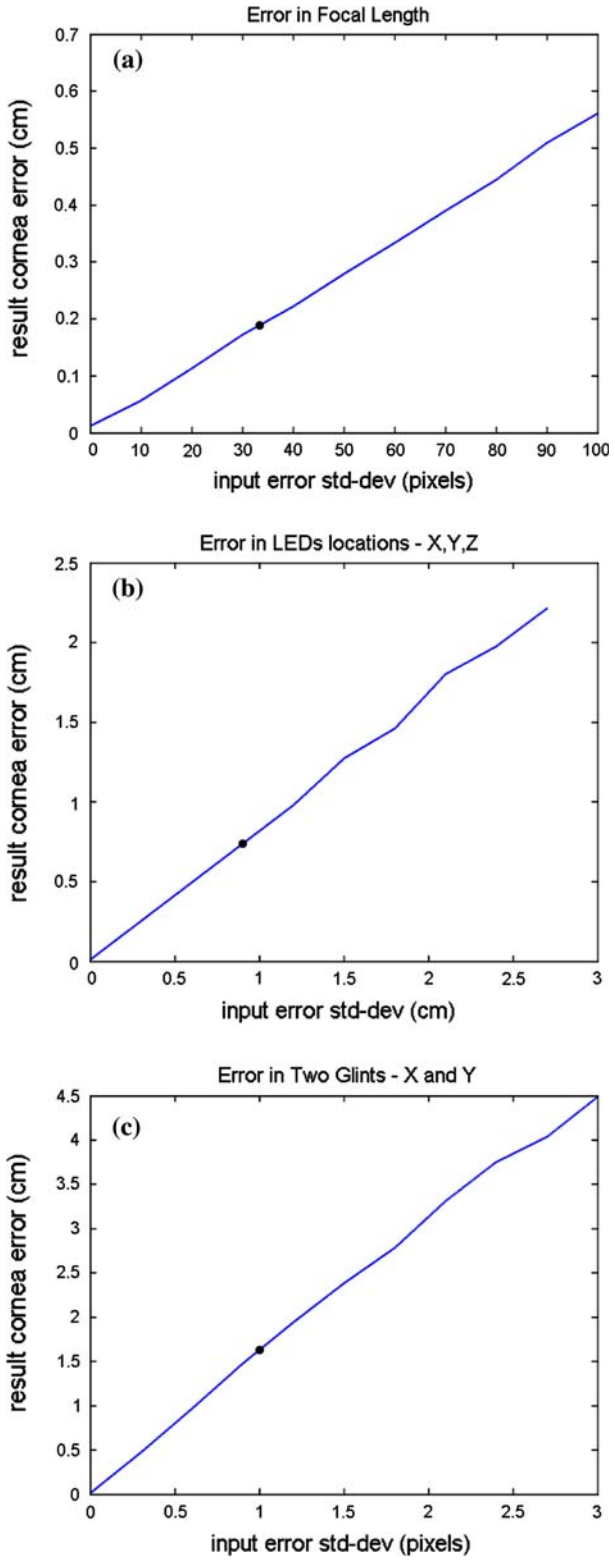

Fig. 8 The influence of error in different parameters on the 3D cornea reconstruction. Note that the maximal standard deviation presents very extreme cases. The *dot* in each graph marks the more typical standard deviation. **a** Influence of the error in focal length. **b** Influence of the error in the 3D LEDs locations. **c** Influence of the error in 2D detection of the image glints

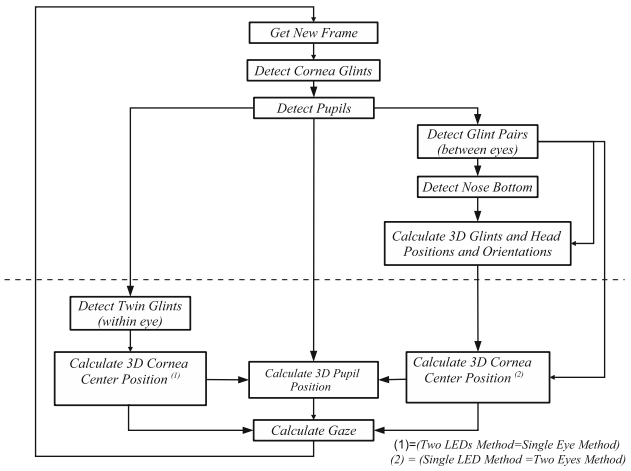

Fig. 9 The system flow chart, showing the different stages of the process, when both approaches are used. The computations concerning both eyes appear above the *dashed line*. The computations which are done per-eye appear below the *dashed line*

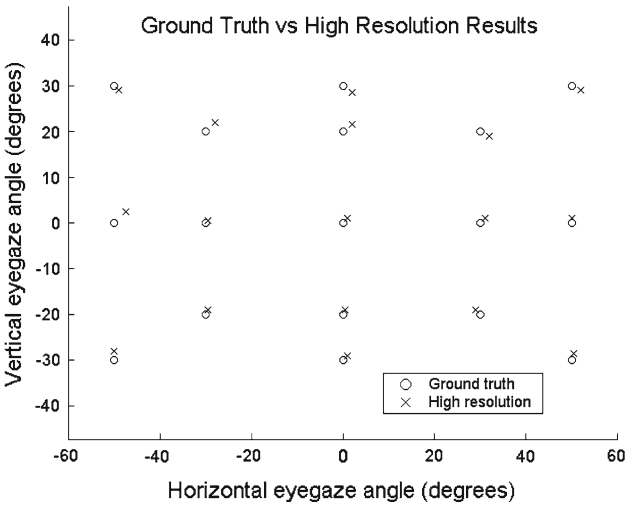

Fig. 10 Ground truth points: *cross*; system results: *circle*

gives a measure of the deviation of d' along the x-axis (respectively the y-axis). In our context, the z-axis is the camera optical axis, so the plane $z = 0$ is parallel to the image plane.

Using this measure, we conducted the following experiment. A group of 20 test subjects were asked to gaze in a set of known directions. This was achieved by having the people looking at a set of points located on a grid of known dimensions on a wall in front of them, while their head location was fixed by a headrest. The system computed their gaze direction. We compared the results between the ground truth and the results given the system. Graph 10 gives the ground truth direction versus the average results computed by the system. The average error is 1.2 degree.

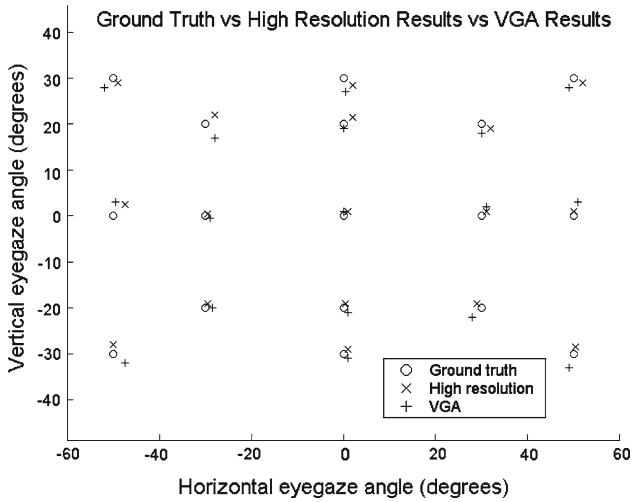

Fig. 11 A group of persons were asked to look at fix points defined as for the experiment with high-resolution camera. The average results are displayed in this graph. The following symbols are used: ground truth points: *cross*; high-resolution camera: *circle*; VGA camera: *plus*. The use of a VGA camera does not compromise significantly the results accuracy. Sometimes, VGA camera-based results turned to be even slightly better than the results obtained with a high-resolution camera. However, we think it is a statistical artifact and not a actual trend

Despite designing our system intentionally to be used even when no user calibration is possible, we elaborated a personal calibration procedure that may be used when this feature is desirable. In that context, each user is required to look at ten predefined directions, while the system keeps the computed gaze directions in memory. Then an affine transformation is computed by least squares for each user, in order to correct the system output and to map it to the ground truth for the calibration direction. Once this transformation is computed, it is applied to every new result the system computes for this person. This simple procedure increases the accuracy of the system, so that the average error drops to half a degree.

The results reported thus far used a high resolution camera. However, the system was also tested with a standard VGA resolution camera. Figure 11 shows no significant loss in accuracy.

4.2 Images

We show sample images produced by the system with the first method, where one can see the detected triangle, defined by the eye centers and the bottom point of the nose. In addition, the gaze line is reprojected onto the images and rendered by white or black arrows, as shown in Figs. 12, 13, 14 and 17.

We also show the system working with two LEDs in images 15 and 16. Finally in Fig. 17, we demonstrate the capability of the system working when the subject wears sunglasses.

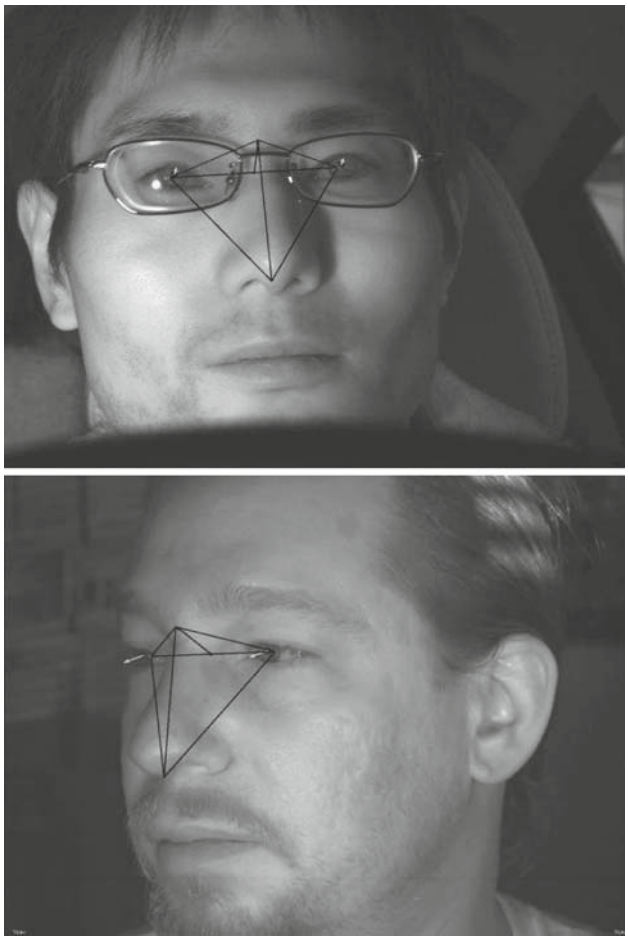

Fig. 12 The detected triangle, eye centers and the nose bottom, together with the gaze line

5 Discussion

We proposed an automatic, nonintrusive eye-gaze system. The system is comprised of two cooperating and complementing algorithms.[2]

The first uses an anthropomorphic model of the human face to calculate the face distance, orientation and gaze angle.

As seen in Sect. 2.4, using a single model for all individuals does not introduce large errors into the gaze direction computation.

The second algorithm requires only a single eye to calculate the gaze.

All computations are done without requiring any user-specific calibration.

While the benefits of a calibration-free system allow for a broad range of previously impossible applications, the

[2] Two patents (numbers in Japan: 2005-253778 and 2006-095008) covering the geometric aspects, the features detection and gaze detection from a single eye were published.

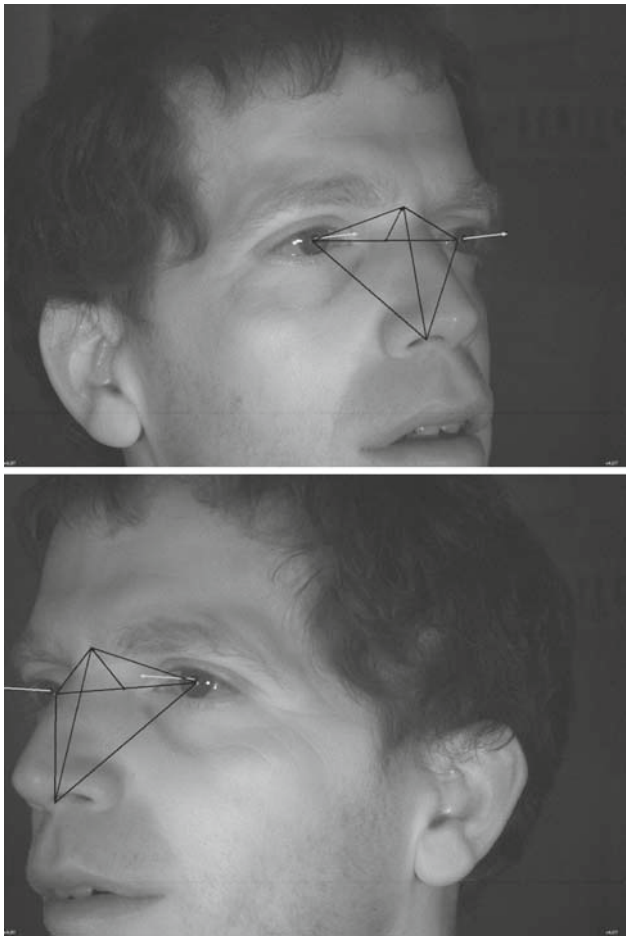

Fig. 13 The detected triangle, eye centers and the nose bottom, together with the gaze line. The detection and reconstruction works even with a wide angle between the subject face and the camera

Fig. 14 The detected triangle, eye centers and the nose bottom, together with the gaze line. The detection and reconstruction are also available when the subjects wears glasses

system design allows for easy plugging of user-specific calibration data, which increase the accuracy even more.

Acknowledgments We are particularly grateful to Dana Shavit and Suzuki Kazufumi for their numerous ideas and support during this research.

Appendix: Computing the intersection of conics in the affine plane

For sake of completeness, we shall shortly recall one way for computing the solutions of the system similar to the one defined by Eqs. 12 and 13. For more details, see [13]. Consider first two uni-variate polynomials with complex coefficients $f, g \in \mathbb{C}[x]$. The *resultant* gives a way to know if the two polynomials have a common root. Write the polynomials as follows:

$$\begin{cases} f = a_n x^n + \cdots + a_1 x + a_0 \\ g = b_p x^p + \cdots + b_1 x + b_0 \end{cases}$$

The resultant of f and g is a polynomial r, which is a combination of monomials in $\{a_i\}_{i=1,\ldots,n}$ and $\{b_j\}_{j=1,\ldots,p}$ with coefficients in $\mathbb{Z}$, that is $r \in \mathbb{Z}[a_i, b_j]$. The resultant r vanishes if and only if either a_n or b_p is zero or the polynomials have a common root in $\mathbb{C}$. The resultant can be computed as the determinant of a polynomial matrix. There exist several matrices whose determinant is equal to the resultant. The best known and simplest matrix is the so-called Sylvester matrix, defined as follows:

$$S(f,g) = \begin{bmatrix} a_n & 0 & 0 & \ldots & 0 & b_p & 0 & 0 & \ldots & 0 \\ a_{n-1} & a_n & 0 & \ldots & 0 & b_{p-1} & b_p & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ldots & \vdots & \vdots & \vdots & \ddots & \ldots & \vdots \\ a_0 & a_1 & \ddots & \ldots & \vdots & b_0 & b_1 & \ddots & \ldots & \vdots \\ 0 & a_0 & \ddots & \ldots & \vdots & 0 & b_0 & \ddots & \ldots & \vdots \\ \vdots & 0 & \ddots & \ldots & \vdots & \vdots & 0 & \ddots & \ldots & \vdots \\ \vdots & \vdots & \ddots & \ldots & \vdots & \vdots & \vdots & \ddots & \ldots & \vdots \\ 0 & 0 & 0 & \ldots & a_0 & 0 & 0 & 0 & \ldots & b_0 \end{bmatrix}$$

Fig. 15 When the head rotation with respect to the camera is large, the gaze is detected from a single eye

Fig. 16 The gaze is detected from a single eye, with all subjects, even if the subject wears glasses. In that sequence, in order to demonstrate gaze detection from a single eye (two LEDs method) in various contexts, we did not perform either nose detection or 3D face orientation computation. Note that while the gaze is detected independently from each eye, the results are very similar in the two eyes. This also demonstrates the robustness of the algorithm

Therefore, we have:

$$r = \det(Syl(f, g)).$$

In addition to this expression which gives a practical way to compute the resultant, there exists another formula of theoretical interest:

$$r = a_n b_p \Pi_{\alpha,\beta}(x_\alpha^f - x_\beta^g),$$

where x_α^f are the roots of f and x_β^g are those of g. This shows that the resultant vanishes if and only if the two polynomials have a common roots. It can be shown that the resultant is a polynomial of degree np.

An important point is that the resultant is also defined and has the same properties if the coefficients of the polynomials are not only numbers but also polynomials in another variable. Hence, consider now that $f, g \in \mathbb{C}[x, y]$ (polynomials with complex coefficients in two variables) and write:

$$\begin{cases} f = a_n(x)y^n + \cdots + a_1(x)y + a_0(x) \\ g = b_p(x)y^p + \cdots + b_1(x)y + b_0(x) \end{cases} \quad (17)$$

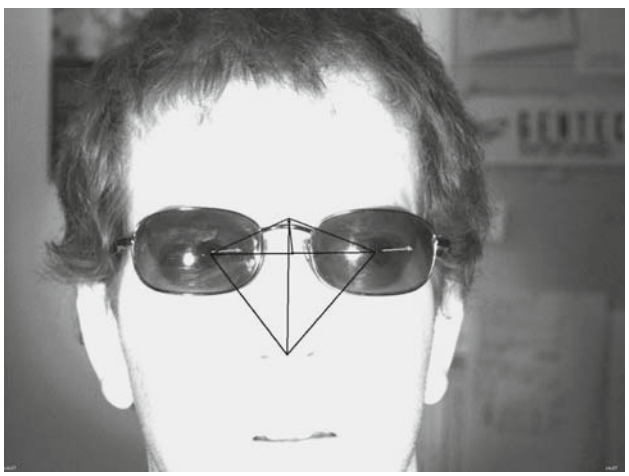

Fig. 17 The system is robust even when the subject wears sunglasses, thanks to histogram-based sunglasses detection

where $a_i(x)$ and $b_j(x)$ are polynomials in $\mathbb{C}[x]$. The question is now the following: given a value x_0 of x, do the two polynomials $f(x_0, y)$ and $g(x_0, y)$ have a common root? The answer to this question is based on the computation of the resultant of f and g with respect to y [i.e. using the presentation given by (17)]. This is a uni-variate polynomial in x, denoted by $r(x) = \text{res}(f, g, y)$.

The resultant can be used in many contexts. For our purpose, we will use it to compute the intersection points of two conics, which are two planar algebraic curves. Consider the curve $\mathscr{C}_1$ (respectively $\mathscr{C}_2$) defined as the set of points (x, y) which are roots of $f(x, y)$ [respectively, $g(x, y)$]. We want to compute the intersection of $\mathscr{C}_1$ and $\mathscr{C}_2$. Algebraically, this is equivalent to computing the common roots of f and g. Therefore, we use the following procedure:

- Compute the resultant $r(x) = \text{res}(f, g, y) \in \mathbb{C}[x]$.
- Find the roots of $r(x)$: $x_1, \ldots, x_t$
- For each $i = 1, \ldots, t$, compute the common roots of $f(x_i, y)$ and $g(x_i, y)$ in $\mathbb{C}[y]$: $y_{i1}, \ldots, y_{ik_i}$.
- The intersection of $\mathscr{C}_1$ and $\mathscr{C}_2$ is therefore: $(x_1, y_{11}), \ldots, (x_1, y_{1k_1}), \ldots, (x_t, y_{t1}), \ldots, (x_t, y_{tk_t})$.

In our context, the resultant r is a polynomial of degree 4 and so $t \leq 4$ and $k_i \leq 2$. To complete the picture, we only need to mention an efficient and reliable way to compute the roots of a uni-variate polynomial. The algorithm that we will describe is very efficient and robust for low degree polynomials. Given a uni-variate polynomial $p(x) = a_n x^n + \cdots + a_1 x + a_0$, one can form the following matrix, called the *companion matrix of* p:

$$C(p) = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \\ -a_0/a_n & -a_1/a_n & -a_2/a_n & \ldots & -a_{n-1}/a_n \end{bmatrix}$$

A short computation shows that the characteristic polynomial of $C(p)$ is equal to $-\frac{1}{a_n} p$. Thus, the roots of p are exactly the eigenvalues of $C(p)$. This provides one practical way to compute the roots of a uni-variate polynomial.

References

1. Farkas, L.: Anthropometry of the Head and Face. Raven Press, New York (1994)
2. Faugeras, O., Luong, Q.: The Geometry of Multiple Images. MIT Press, Cambridge (2001)
3. Gee, A., Cipolla, R.: Estimating gaze from a single view of a face. In: IAPR 12th International Conference on Pattern Recognition (1994)
4. Gee, A., Cipolla, R.: Non-intrusive gaze tracking for human–computer interaction. In: International Conference on Mechatronics and Machine Vision in Practice (1994)
5. Glenstrup, A., Engell-Nielsen, T.: Eye Controlled Media: Present and Future State. University of Copenhagen, DK-2100 (1995)
6. Hartley, R., Zisserman, A.: Multiple-view Geometry. Cambrigde University Press, London (2000)
7. Horprasert, T., Yacoob, Y., Davis, L.: An anthropometric shape model for estimating head orientation. In: 3rd International Workshop on Visual Form, Capri, Italy (1997)
8. Huynh, D.Q.: The cross ratio: a revisit to its probability density function. In: British Machine Vision Conference, vol. 1, pp. 262–271, 11–14 September (2000)
9. Ji, Q., Yang, X.: Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. Real-Time Imaging **8**, 357–377 (2002)
10. Kaminski, J.Y., Knaan, D., Shavit, A., Teicher, M.: Head orientation and gaze detection from a single image. In: Proceedings of International Conference of Computer Vision Theory and Applications (2006)
11. Ohno, T., Mukawa, N., Yoshikawa, A.: Freegaze: A gaze tracking system for everyday gaze interaction. In: Symposium on Eye Tracking Research and Applications (2002)
12. Perez, A., Cordoba, M.L., Garcia, A., Mendez, R., Munoz, M.L., Pedraza, J.L., Sanchez, F.: A precise eye-gaze detection and tracking system. In: 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (2003)
13. Sturmfels, B.: Solving Systems of Polynomial Equations. American Mathematical Society, Providence (2002)
14. Ueno, K.: Algebraic Geometry 1. American Mathematical Society, Providence (1999)
15. Wang, J.G., Sung, E., Venkateswarku, R.: Eye gaze estimation from a single image of one eye. In: 9th IEEE International Conference on Computer Vision (2003)
16. Yilmaz, A., Shah, M.A.: Automatic feature detection and pose recovery for faces. In: The 5th Asian Conference on Computer Vision, 23–25 January (2002)