# Διατύπωση ή περιγραφή του προβλήματος

### How to

1. Describe how things should work.

Σημασία και χρήσεις eye tracking. Ανάγκη πρόσβασης σε εργαλεία χαμηλού κόστους χωρίς πρόσθετο hardware (μόνο με χρήση απλής κάμερας).

1. Explain the **problem** and state why it matters.

Οι υπάρχουσες υλοποιήσεις είναι ακριβές, χρησιμοποιούν πρόσθετο ή και invasive hardware. Οι υπάρχουσες υλοποιήσεις low cost σε απλή κάμερα δεν έχουν μεγάλη ακρίβεια και βασίζονται στην υπόθεση ότι το κεφάλι μένει σταθερό (χωρίς περιστροφή ή μετατόπιση), ότι υπάρχουν καλές συνθήκες φωτισμού, ότι η κάμερα βρίσκεται σε συγκεκριμένη θέση σε σχέση με την οθόνη. Επίσης χρειάζονται μεγάλη διαδικασία calibration.

1. Explain your **problem's** financial costs.
2. Back up your claims.
3. Propose a solution.
4. Explain the benefits of your proposed solution(s).
5. Conclude by summarizing the **problem** and solution.

### O2

Driver diversion is the most common cause of focus deviation from the lane, and it can put drivers, riders, and pedestrians in grave danger. Eye monitoring methods have been gradually applied in driving fatigue-warning systems as a result of research.

▪ However, there are many limitations, including dependable real-time performance, high precision, device availability, and a lightweight and non-intrusive device.

▪ Even though many experiments on gaze engagement have been published, the efficiency of these approaches in smart interactive settings under real-world conditions is still lacking.

▪ A deep learning-based gaze estimation technique has been considered to solve this challenge, with an emphasis on Convolutional Neural Networks (CNN) based methods.

### O3

### SD1

These examples show that despite the variety of areas in which eye-tracking systems are used, often low-cost tools with low tracking accuracy are used to implement the technology. This suggests a need for low-cost eyetracking devices comparable in quality to laboratory equipment, which would allow scientists from various fields to obtain results with high accuracy. It is also important that the source code be open, allowing customization of the software. The main limitation of modern eye-tracking systems is their price. Equipment with an accuracy of about 1−2 ◦ costs several thousand dollars. Here, we develop a low-cost eye-tracking system based on deep learning methods.

# Επισκόπηση πεδίου

### How to

1. Narrow your topic and select papers accordingly.
2. Search for **literature**.
3. Read the selected articles thoroughly and evaluate them.
4. Organize the selected papers by looking for patterns and by developing subtopics.
5. Develop a thesis or purpose statement.
6. **Write** the paper.
7. **Review** your work.

### SD3

Specifically, eye detection and localisation methods that are categorised as shape-based, since they exploit the shape and contours of the open eye, typically aim to fit a geometrical model to the eye region [21-29,29-36]. These methods approach the problem of eye detection either by fitting the iris or pupil region with a simple circular [21-30] or elliptical model [29,31,32], or by the use of more complex models [33-36], which are more computationally demanding than their simpler counterparts but allow for detailed modelling of the eye region by including components such as the eyelids [33-36] and eye corners [36]. Both subcategories of shape-based methods typically require high contrast images that permit reliable feature extraction for shape fitting [21,23,24,26,27,32,33,35], together with the initialisation of the shape models close to the eye region for successful localisation [28] or the acquisition of close-up eye region images within which the model-fitting is carried out directly [29,32,35]. Appearance-based methods, in contrast, are independent of the actual geometry since these exploit the photometric properties of the eye region for detection [37-51]. Eye detection by these methods follows different approaches, such as via machine learning [37-44] or template-matching techniques [45-51], which preserve the spatial and intensity information of the eye image pixels. Most appearancebased methods that rely on machine learning require the collection of a sizeable set of training data under the expected tracking conditions, upon which the performance of eye detection is subsequently contingent [44]. Furthermore, methods that rely on image templates for eye detection are often inherently limited by rotation and scale constraints [48]. Feature-based methods, on the other hand, seek to identify specific informative local features of the eye region that may be less susceptible to illumination changes and variations in viewpoint [52-65]. Detection of these local features may either exploit the grey-level differences at the feature boundaries, such as the limbus boundary [52-55], or alternatively the dark and distinctive colour of the pupil [56-65]. In order to reduce the number of eye candidates that may be captured by wide field-of-view imaging hardware and which may exhibit similar features to the eyes, feature detection is often performed within close-up eye region images [63-65]. Different methods for eye detection and localisation are summarised in Fig. 2 and illustrated in Fig. 3.

### SD4

2.1. Appearance-based methods In recent years more and more applications have been developed to analyze human behavior in everyday situations. To do this, it is necessary to monitor the eye movements in uncontrolled scenarios where it is impossible to adjust the lighting conditions, perform a calibration or request the user's assistance. In this context, it is critical to design very robust methods for the different types and qualities of the images. Fortunately, for most applications very high accuracy is usually not required. For instance, in environmental control or eye typing, where only a few buttons need to be activated, it may be more important to reduce costs by using web cameras, allowing easy and flexible hardware configurations, and avoiding the use of lighting systems and feature detection algorithms. On the basis of this new approach, several papers [27–30] have presented methods that work with low-resolution images in different environmental conditions in which appearance-based methods seem to be a promising option. These methods address the gaze estimation problem by learning a mapping function directly from eye images to gaze directions. As input, they use all eye regions pixel values as highdimensional feature vectors for estimating gaze directions [31]. The output, gaze direction, can be represented as the coordinates (x y , ) on the screen where the gaze falls or the rotation angles of the eye with respect to the head position. For low quality images, this is a great advantage compared to the techniques employed by feature-based methods, which have to segment and analyze geometrically derived eye features from high-resolution observations as will be seen in the next section. The mapping function allows to relate the raw input image with the coordinates of the gaze direction. These functions do not address any particular model, but are designed ad-hoc and are trained with eye images of known gaze direction using various regression techniques, including neural networks [32–34], local interpolation [35,36], or Gaussian process [37,38]. Its formulation depends on the regression technique followed. Fig. 1 shows an example where a convolutional neural network (CNN) is used as a mapping function. These approaches make the system less restrictive, and even though the precision is not good enough for certain applications, they are very robust even when they are applied to relatively low-resolution cameras or under natural illumination, such as with a phone or computer applications or human computer interaction. The main problem of these methods is that the appearance of an eye Fig. 1. Architecture of the CNN used as mapping function to predict gaze direction. (From Park et al. [27]). A.J. Larrazabal, et al. Computers in Biology and Medicine 108 (2019) 57–66 58depends not only upon gaze direction but also upon the head poses, imaging conditions and even on the identities of subjects, making it necessary to generate a person-specific training. In addition, due to the high dimensional feature vectors that must be mapped into the gaze directions, thousands of individual training samples are required to calculate the mapping coefficients. To overcome these limitations, Sugano et al. [39] propose a learning-by-synthesis approach to appearance-based gaze estimation using a large dataset that contains diverse people, head poses, and gaze directions. Also to avoid the need of a person-specific training, Lu et al. [40] extract more advanced eye features, which help to learn a personindependent relationship between eye gaze change and eye appearance variation. On the other hand, Schneider et al. [41] perform embedding for each person in the training set and then learn a linear transformation that maps out the individual, subject-dependent manifolds avoiding the need of individual calibration. Despite the recent research progress in the field of computer vision, estimating human gaze directions from only eye appearance is still an open challenge. The performance of appearance-based methods generally depends on the quality and diversity of the training data and generalization ability of the regression algorithm. Moreover, their accuracy is not high enough for clinical uses. For these reasons, appearance-based methods can be ruled out for devices designed for this purpose. 2.2. Feature-based methods Methods using extracted local features such as contours, eye corners, and eye reflections, called feature-based methods, are the most popular approach for gaze estimation. These methods use geometrically derived eye features from high-resolution eye-images captured by zooming in the user's eyes (See Fig. 2). Once the features are extracted, the connection between the gaze directions and them can be modeled in various ways. Besides, depending on whether they are based on eye geometry or not, these methods can be divided into two main groups: 2D mapping-based gaze estimation methods and 3D model-based gaze estimation methods. The 3D model-based methods [42,43], directly compute the 3D gaze direction vector from the eye features based on a geometric model of the eye. Then, the point of gaze is estimated by intersecting the gaze direction with the object being viewed, i.e a computer monitor. In order to calculate the center of the cornea and the eye vector, these models require accurate estimation of many user-dependent parameters such as cornea radii, angles between visual and optical axes, the distance between the cornea center and pupil center, among others. To understand why these parameters should be estimated, and which complex hardware calibration should be made during initial setup, the model proposed by Guestrin et al. [44] will be developed. This example is also a good basis for understanding model-based methods. The model and their parameters are shown in Fig. 3. Considering a ray that comes from the light source Ii, reflects at a point qi j, on the corneal surface, which is modeled as a convex spherical mirror of radius R, passes through the nodal point of the camera oj, and intersects the camera image plane at a point ui j, , the next two equations can be formulated: q () =+ − o ou k for some k ij j q ij j ij , , q ij (1) || || q c ij − = R (2) In addition, based on the beam reflection laws, two more equations can be raised for these points. (i o q oco i j −× − −= ) ( )•( ) 0 ij j j (3) − −⋅ − = − −⋅ − i q q co q o q q ci q ( )•( ) || || ( )•( ) || || i ij ij j ij j ij ij i ij (4) In the same way, considering a ray that comes from pupil center p, refract at the point rj on the corneal surface, passes through the nodal point of camera oj, and intersects the camera image plane at a point vij, two more equations can be obtained. rij j r j j ij =+ − o ov k for some k , , ( ) r j (5) || || r c ij − = R (6) Then, applying beam refraction laws, the following equations are derived where n1 and n2 are the refraction index of the aqueous humor and cornea combined and of air respectively. (r o copo jj j j − ×− − = ) ( )•( ) 0 (7) −×− ⋅− = −× − − n n r c pr o r r c o r pr ||( ) ( )|| || || ||( ) ( )||•|| || j j j j j jj j 1 2 (8) Finally, considering K as the distance between the pupil center and the center of corneal curvature leads to: || || p c − = K (9) By means of solving the proposed system of equations for c and p, the optic axis of the eye in the space can be reconstructed as the line defined by these two points. It is important to note that to solve these equations, all the subject-specific parameters (R, K and n1) have to be known. In general, if only one camera is available, they are obtained by the calibration process -detailed below-. Also, the angle between the optic axis and visual axis must be calculated and is usually done during the calibration procedure. This parameters also rely on metric information requiring camera calibration and exact knowledge of the light sources and monitor position. These values may be directly measured once during the first setup but, to achieve a high accuracy, the eye parameters need to be estimated independently for each individual, making that a previous calibration step cannot be omitted. The 3D model-based approaches can handle head movements in a robust manner with high accuracy but involving this relatively complex initial setup. They need to use at least a single camera with multiple calibrated light sources [44] or stereo cameras [45–47]. Even so, for some clinical diagnoses, it is important to be able to differentiate between oculocephalic and pure eye movements, so calculating the absolute position of the gaze is not always useful. Furthermore, regardless of the model complexity, the calibration might be only simplified, but Fig. 2. Features from high-resolution eye-images (from Park et al. [43]). Fig. 3. Schematic representations of the eye, a camera, and a light source (from Guestrin et al. [44]). A.J. Larrazabal, et al. Computers in Biology and Medicine 108 (2019) 57–66 59not avoided at all. In some works, to avoid the calibration process, a very simplified eye model is used. While it reduces calibration times and complexity, the accuracy obtained also greatly decreases. On the other hand, the 2D mapping approaches [23,48,49] are based in finding a mapping function from 2D feature space like PupilCenter-Corneal-Reflections (PCCR), contours, etc. to gaze point such the computer screen coordinates. That function avoids the need for the direct measurement or estimation of the eye model parameters throughout the system setup. Instead, they are implicitly included in the learning of the mapping function simplifying the setup process itself. The same happens with the camera calibration process and the system geometry determination. Different features are used as inputs to the mapping function depending on the application and the image conditions. Mostly, they can be further divided into active light techniques such as PCCR or passive light techniques such as shape-based methods, depending on whether they require external light sources to detect eye features. In the recent years, eye tracking applications using webcams under natural illumination have gained highly relevance in the community. In particular, passive image-based algorithms for eye localizing and tracking in the visible spectrum have been researched over the last years [27,50,51]. These algorithms propose the search for some features like iris or pupil center. For the purpose of iris tracking, the limbus, which is the boundary between the sclera (normally white) and iris (comparatively dark) is optically detected and tracked. Pupil tracking is similar to iris tracking except that a smaller boundary between iris and pupil is used for relative measurement. Although without active illumination it is easier to segment the limbus due to the higher contrast between the iris and the sclera compared to the contrast between the pupil and the iris, pupil tracking has a lot of advantages. The pupil, which is much less covered by the eyelids than the limbus, enables vertical tracking. In addition, the sharper edge between the pupil and the iris provides a higher resolution. Various iris and pupil center localization methods have been reported in the literature [52,53]. Several treat iris or pupil center localization as a circle detection or ellipse fitting problem [54–57]. Depending on the viewing angle, both iris and pupil appear elliptical and consequently can be modeled by different shape parameters. Simple ellipse models consist of voting-based methods [58,59] and model fitting methods [60,61]. Once the iris center has been successfully localized, regression-based methods can be used for finding the corresponding gaze points on the screen. Since these methods directly map the eyes iris center or pupil center location to a target plane such as the monitor screen, the accuracy and robustness of the center localization significantly affect the performance of gaze tracking. For example, detection has some problems when the iris moves toward the corners or when the upper and lower boundaries of the iris are occluded by the eyelids and eyelashes, leading to gaze estimation errors. On the other hand, for applications like clinical research, where experiments are performed in a doctor's office, it is not a problem to have infrared lighting, and thus active methods would be a better option. PCCR is the most common approach for feature-based gaze estimation methods. When a light source (usually infrared) illuminates the eyes at different layers, the boundaries between the lens and the cornea act as convex mirrors and produces some reflections or virtual images, which are called corneal reflections or Purkinje images. In particular, the Purkinje image formed by the reflection of the outer surface of the cornea, called the first Purkinje image, is known as glint. The glint is the brightest and easiest reflection to detect and track. The PCCR technique uses the vector formed by the subtraction between the estimated center of the pupil and one or more near infrared (NIR) corneal reflections to estimate the gaze direction [49,62]. To compute the pupil-glint vector, the pupil center must also be extracted from the image. As it was already mentioned, different techniques are avaible for doing this but, with active illumination, the bright pupil-dark pupil method (BP-DP) is one of the most widely used for determining the accurate location of the pupil. When a light source is placed collinearly to the optical axis of the camera, most of the light is reflected back to the camera and the eye image shows a bright pupil. Conversely, when a light source is located away from the camera's optical axis, the image shows a dark pupil. Therefore, eye trackers with active IR illumination can use the difference between dark and bright pupil images by synchronously switching between the two light sources. This technique is very simple and robust in controlled conditions [11]. Besides that, the detection of the corneal reflections requires a narrow field of view (FOV) camera (long focal length) since the reflections are in general very small. Therefore, these systems work with high-resolution eye images captured by zooming in on movement-restricted users. Under these conditions, eye features can be easily and robustly extracted, this being an advantage over other methods. These reported techniques are widely used and achieve really good results, but they have two major issues. First, because the mapping function is different for each person and for each system configuration, it is necessary to perform a tedious calibration procedure before each test to obtain the necessary parameters. In a typical calibration procedure, a set of visual targets such as those shown in Fig. 4, is presented to the user who normally has to stare to the computer for a period while the corresponding measurement is being done. Afterwords, from these correspondences, a mapping function is calculated. The second drawback is that once the calibration has been performed, the person's head must remain motionless. Otherwise, there will be large errors between the actual and estimated directions. To avoid these errors head restraint systems are often used, and the calibration process is repeated every time movements are observed in the patient. Fig. 5 shows an active feature-based system, and the restraint system it uses to prevent head movement. With this device, Hernandez et al. [23] achieve an accuracy of less than 0.4∘ , reported as one of the minimum reaches in the literature. A number of efforts are being made to minimize these shortcomings. 2D mapping methods assume that the mapping function have a particular parametric form such as a polynomial or a non-parametric form such as a neural networks whose coefficients have no physiological or physical meaning. Polynomial interpolation is one of the main tools for parametric mapping functions, mainly due to its simplicity of execution and the good quality of the result obtained from it. In this case, the x and y gaze coordinates are estimated by means of a polynomial function. For example, a second order polynomial transformation is defined as: =+ + + + + =+ + + + + x a ax ay axy ax ay y b bx by bxy bx by , c e e e e e e c e e e e e e 12 3 4 5 2 6 2 12 3 4 5 2 6 2 (10) where (x y c, )c is the coordinate of the point on the screen where the gaze Fig. 4. Example of the calibration points. A.J. Larrazabal, et al. Computers in Biology and Medicine 108 (2019) 57–66 60falls, (x y e, )e is the coordinate of the pupil-glint vector and ai; bi are the polynomial coefficients. These coefficients are calculated in the calibration procedure. During this procedure, the patient is asked to stare at a set of known targets, while a set of corresponding points are obtained. For example, for a 6-point calibration procedure, 6 corresponding points are obtained (x y ci, ) ci ; (x y ei, ) ei with i = … 1,2, 6 and a system of 12 equations is generated to calculate the polynomial coefficients ai; bi, by applying the least squares estimation procedure, that is, minimizing the quadratic error E2 between the estimations and the calibration points coordinates. The higher is the order of the polynomial mapping function, the greater will be the number of calibration points needed to calculate all coefficients. In Equation (11) the quadratic error function for N calibration points is displayed. = ∑ −+ + + + = ∑ −+ + + + = = E N x a ax ay ax y E N y b bx by bx y [ ( ..)] [ ( ..)] x i ci ei ei ei ei y i ci ei ei ei ei 2 1 12 3 4 2 2 1 12 3 4 2 (11) Mimica et al. [63] use a second order polynomial to minimize the number of calibration points required comparing to those required by a higher order polynomial. Cerrolaza et al. [64,65] carried out a study on the potential effect of the order and systematic inclusion of all polynomial terms, on the accuracy and robustness of the gaze tracker. For this, a real VOG system with different configurations was used. The authors point out that the gaze estimation accuracy of a gaze tracking system is not noticeably increased with the enhancement of polynomial order or with more complete mathematical expressions due to the factors of head motion, and calculation method of the pupil-glint vector. The choice of the mapping function determines not only the accuracy of the system but also the head movement tolerance and the calibration time. Therefore, when linear regression solution methods are applied to solve the mapping function, a second-order linear polynomial is the most used due to its advantages of less calibration markers and better approximation effect. Alternatively, Baluja et al. [32] first proposed a method using a simple artificial neural network (ANN) to calculate a non-linear mapping function. First, they mapped images of only the pupil and cornea as the inputs to ANN to the coordinates of the gaze point as the outputs. Then, they included the total eye socket as an input to improve the system accuracy (about 1.5∘ ). In addition, Zhu and Ji [66] utilize generalized regression neural networks to estimate the gaze direction. For this purpose, 6 pupil and glint parameters were used as inputs to the calibration procedure. The parameters were chosen in such a way that they represent eye and head movements and remain relatively unchanged for different people. Therefore, even though the accuracy acquired is not good enough (about 5∘ ), it is a free calibration process and head movements are allowed. In a similar way, Gneo et al. [67] utilize multilayer neural feedforward networks to calculate gaze point coordinates based on pupilglint vectors. In order to minimize the number of output neurons, they use one separate network with the same input for each gaze coordinate (x y , ). The reported results were competitive with high accuracy (about 0.6∘ ). More recently, Wang et al. propose in Ref. [49] an improved ANN based on direct least squares regression to calculate the mapping function between pupil-glint vectors and actual gaze points. They combine the advantages of both methods: the high speed of direct least squares regression and the high accuracy of ANN. They achieved a good accuracy (about 0.4∘ ) in a head-mounted device which can be seen in Fig. 6. Thus, as it was pointed out before, the choice of the model depends on multiple factors: required accuracy, hardware cost, image quality/ eye region resolution, available information in the image (e.g., glints), and configuration flexibility. For instance, feature-based methods accuracy may decrease when model assumptions are violated. In some applications such as clinical research or disease diagnosis, where it is possible to control the illumination conditions, the camera's quality, and the system settings, these methods achieve a really high accuracy which is critical to investigate imperfections in the oculomotor system [68]. Moreover, despite the fact that mapping methods provide little information about the intrinsic behavior of the system, they are much simpler to construct than the model-based methods and do not require additional hardware calibration, which makes setup much faster for the system user. That is why, most commercial gaze tracking systems use 2D mapping features-based methods with IR camera and active IR illumination, as it is shown in Fig. 7, to achieve the highly accurate performance of gaze estimation. 2.3. Head movements So far we have talked about gaze tracking as a process that belongs exclusively to the eyes, but it is known that the gaze is a product of two contributing factors, the head pose (position and orientation) and the eyeball orientation. A person can change gaze direction by rotating the eyeball while keeping the head stationary; similarly, a person can change gaze direction by moving the head while keeping the eye stationary relative to the head. Usually, a person moves the head to a comfortable position before Fig. 5. Example of a gaze tracker with a restraint system (from Hernandez et al. [23]). Fig. 6. Example of a head-mounted gaze tracker system (from Wang et al. [49]). A.J. Larrazabal, et al. Computers in Biology and Medicine 108 (2019) 57–66 61orienting the eye. Head pose, therefore, determines the coarse-scale gaze direction while the eyeball orientation determines the local and detailed gaze direction. For example, when the target is located over 20∘ of the field of view, it is much more comfortable to rotate the head than to rotate the eyeball. While in the previous section we reviewed the state-of-the-art in gaze estimation systems, focusing on tracking eye movement, the problem of ensuring the invariance to head movements is also an important and a challenging research topic. In almost all applications, people move their heads while they are using a gaze tracker. For this reason, for an accurate gaze estimation, it is necessary to (either directly or implicitly) model both head pose and eye rotation. As we pointed out above, appearance-based methods have been developed in order to be applied in environmental conditions without the possibility of monitoring user conditions. This user freedom movement situation requires the method to be robust to changes in head position. For solving this problem two possibilities can be held: learning generic gaze estimators from large amounts of head pose-independent training data or adding head pose information to the eye images. In particular, Lu et al. in Ref. [69] address the head motion problem by synthesizing new training images for different head poses from those already seen in estimation, while in Ref. [70] they perform the gaze estimation by assuming a fixed head pose and then compensating for the estimation biases caused by the head pose using a head pose tracker. Conversely, Zhang et al. in Ref. [27] used a multimodal convolutional neural network to learn a mapping function from both the head poses and eye images to the unique gaze directions. Lai et al. in Ref. [71] also combine the eye image information with head pose tracking selecting features by means of the neighborhood-based regression algorithm. Although these results outperform the appearancebased methods state of the art, the accuracy achieved is still quite poor what that required for clinical applications. Regardless of the eye tracker method applied, other researchers have attempted to measure the head movements or head position directly, and to use that information to correct the gaze measurements. These systems usually include two or more cameras and use a complex facial model to track the movement of the face [72,73]. In order to track the face from these models, the FOV of the tracking camera has to be large enough to cover the entire user's head. This is not a problem for daily applications which do not need such a high accuracy like in Ref. [74], but, when feature-based methods are applied, these restrictions make it difficult to locate the small eye features and result in less accurate gaze tracking. To overcome this drawback, some works have proposed the use of two cameras in combination with pan and tilt mechanisms that allow freedom of person motion while maintaining the feature method accuracy. In Ref. [75], Hennessey et al. proposed a system that rotates an eye tracker with a narrow-angle camera using pan and tilt servo motors, while in Ref. [76] Cho et al. propose a binocular eye gaze tracking system that, using pan, tilt and zoom movements, continue to track the eyes with a narrow camera while the user moves his head freely in depth. Then both estimate the POG by a 2D mapping function modified with the depth of the eyes. All these methods work relatively well but are very complex, expensive and, most notably, slow. These limitations restrict its use so that in practice, head pose information is rarely used directly in the gaze models. It is more common to incorporate this information implicitly either through the mapping function (regressionbased method) or through the use of reflections on the cornea (3D model-based approaches). Apart from that, the 3D model-based methods are the most robust to head pose changes and they can obtain the head pose invariance through various hardware configurations and prior knowledge of the geometry and cameras. Guestrin et al. [44] presented a general study for PCCR covering all the possible system configurations in terms of number and positioning of IR light sources and cameras. In that work the authors claimed that using only one camera with two light sources is the simplest configuration that allows for both the estimation of POG and free head user movements. To estimate the POG with this system configuration, it is necessary to make a subject-specific calibration procedure that requires the subject to fixate on multiple points. To avoid the need of calibration, an additional camera is also necessary. Using one camera and one light source the POG can be estimated only if the head is completely stationary. This restriction is shared with the 2D regression methods, which assume static head conditions. In general, gaze estimation systems that use one camera and one light source assume that the head movements are negligible. Therefore, it should be noted that the only video oculography method that allows large head movements while maintaining good accuracy are some 3D model-based methods. But as a drawback, they require a really complex and calibrated system setting, difficulting their use in common applications. In addition, the movement of the head is taken into account implicitly, and it is not possible to differentiate between oculocephalic and purely ocular movements. This is another reason why the eye location algorithms found in commercially available eye trackers use the 2D regression techniques where the 3D eye location is usually unknown and only the relative orientation of the user's eye with respect to the user's head is measured. Particularly, gaze estimation is based on the relative position between pupil and glint. Assuming a static head, methods based on this idea use the glint as a reference point, thus the vector from the glint to the center of the pupil will describe the gaze direction. While contact-free and non-intrusive, these methods work well only for a static head, but even minor head can fail these techniques. In addition, since the pupil and glints are very small, the FOV of the eye tracking camera has to be confined to obtain a high definition eye Fig. 7. Diagram of a standard eye tracker with 2D mapping method. A.J. Larrazabal, et al. Computers in Biology and Medicine 108 (2019) 57–66 62image. This aspect also limits the head so that the eye does not disappear from the FOV and emphasizes the problem of sensitivity to head pose variations, requiring the user to be either equipped with a headmounted device or to use a high-resolution camera combined with a chin rest to limit the allowed head movements. In many clinic applications where tests are conducted for only a few minutes and it is not easy to perform complex system calibrations, those restraint systems are suitable and work very well. Even so, despite the fact that the head movements are restricted, in people with certain neurological diseases, it is possible to observe some involuntary movements and it is very important be able to measure them not only to correct the gaze estimation errors but also because these measurements are indicators of the presence or progress of certain diseases. As it was pointed out before, there are a lot of research works that deal with the problem of obtaining enhanced gaze estimation in presence of large movements and head pose variations for daily applications [74]. But, despite their importance for clinic diagnosis, there are not many studies performing the feasibility of a gaze estimator that considers both the head and eye movements in a really zoomed and high-resolution images with the objective to detect short involuntary head movements. Some works study the way of achieving head pose invariance for 2D regression methods, for example [66] use Generalized Regression Neural Networks (GRNN) instead of polynomial functions to account head implicitly by the gaze mapping function. They also include more parameters as mapping function inputs like glint coordinates and pupil radio to account for the different head motions. In Ref. [77] authors also employed GRNN but with the aim of compensating the errors generated by the non-linear polynomial for different head poses obtaining a gaze estimation robust to head but with much lower spatial gaze resolution. Despite these advances, there are no studies done to quantify these slight movements based only on the zoomed eye image which is necessary for some neurological disease diagnosis.

### SD5

ith several recent publications [16], the overwhelming majority of which are using hardware-based approaches (i.e. IR light sources, high-resolution cameras, multiple cameras and wearable equipment). The focus of the current overview is on approaches working under natural illumination, using a single, remotely located camera. The current subset of gaze estimation methods can be subdivided into two broad categories, i.e. feature-based and appearance-based methods [16]. Feature-based methods use computer vision techniques to extract and track local eye features such as eye centers, corners and contours. The extracted features can be used to derive feature vectors which can be directly related to gaze. According to the approach used for relating the image features with gaze direction, feature-based methods can be further divided into geometric (or model based) and interpolation based (or regression based). Geometric methods [4,17–20] compute directly the gaze direction from the image features based on a geometric model of the eye, while interpolation based methods [21–26] built a mapping function between them, using parametric (e.g. polynomial) or non-parametric forms (e.g. neural networks). Torricelli et al. [21] use image processing algorithms to extract and track the eye features and perform mapping with gaze direction using neural networks. In [22], Zhu et al. detect and track eye centers and corners using subpixel accuracy and a linear interpolation model for inferring gaze coordinates. The works in [17,18] use geometric models which rely upon facial feature tracking for estimating head pose and eye orientation. The feature vectors between eye centers and eye corners are used in [27,23,24] to derive gaze estimation through 2D interpolation mapping. A comparison between a common polynomial mapping function and a geometric model is performed in [28], giving a slight edge to the latter. Ishikawa et al. [4] use Active Appearance Models (AAM) to detect and track facial points and employ an edge-based algorithm for iris refinement. They subsequently employ geometric models to derive gaze direction combining head pose and eyeball orientation. Authors in [29] also use AAM for iris and eyelid tracking in order to derive gaze information. The gaze angle is geometrically defined combining head pose and eyes position information. In the work of Salam et al. [30] the head pose is derived using a 2.5D global AAM, while a multi-texture AAM is used for iris localization. The contribution of each eye to the final gaze direction is weighted depending on the detected face orientation. In general, global appearance-based methods are very robust in detecting the overall rough positions of the facial features. However, as they depend on the convergence of the full model (i.e. by satisfying a minimization function or reaching a maximum number of iterations), they do not ensure localization of each feature with high precision, thereby adversely affecting the gaze estimation accuracy. Chen and Ji in [19] also use a geometric model to localize facial points, manually extract pupil centers and build a 3D gaze estimation model, tailored with personspecific eye parameters. Heyman et al. [20] track the 3D pose of the head using Canonical Correlation Analysis and extract the positions of the irises using blob detection and 4-connected component labeling. Valenti et al. in [25] employ their eye detection approach based on isocenters, to also detect eye corners which are used as anchor points for gaze estimation. Given the vectors between the detected eye centers and eye corners they perform 2D linear mapping to screen coordinates. Their work is extended in [31] where pose 30 E. Skodras et al. / Signal Processing: Image Communication 36 (2015) 29–42estimation and eye localization algorithms are combined so that they complement each other, thus increasing the system's performance. In their work in [32] they combine eye gaze information derived from their proposed eye gaze tracker and a commercial gaze tracker with saliency maps (probability maps representing the likelihood of receiving eye fixations). Shape modeling of the iris methods have been often employed, many of which exploit the fact that when the iris orientation changes, the shape of the iris appears to deform from circular to elliptical. However, for such shape modeling approaches, higher resolution images are required. In [33,34] the gaze direction is inferred by estimating the shape of the iris through ellipse fitting. Active contour tracking using particle filters is used in [35] to built a generic system, working on various setups and conditions, investigating also the lower bound calibration requirements. Appearance-based or holistic approaches incorporate eye information implicitly by using the intensity distribution or filter responses of the eye area. They usually classify gaze according to the appearance of the eye area for each direction. The systems described in [36–40] learn the gaze direction by modeling the corresponding eye appearance. In the work of Schneider et al. [40] several regression techniques are evaluated, modeling the appearance of the eyes (in terms of features such as Histogram of Oriented Gradients and Local Binary Patters) when gazing at different directions, in order to build a calibration-free system. Hansen et al. [41] use an active appearance model of the eye using shape and texture properties to be used with an eye typing interface. Saliency information of the displayed images is aggregated with an appearance-based gaze estimator in [42]. The main drawbacks of holistic approaches is that the appearance of the eyes is significantly affected by the head pose and usually only a limited number of discrete eye directions are modeled. These limitations can preclude their use in many applications, giving rise to the more prevalent use of feature-based approaches.

### S1

Depth-sensor-based methods usually use Kinect to obtain depth information and convert the POR from the eyeball center, which are simple in configuration and allow for natural head movements. Li et al. [15] performed 3D gaze estimation under natural head movements, wherein the eyeball center was computed with the inner eye corner as the anchor point for head pose tracking. The gaze estimation accuracy was 1.38◦–2.71◦. Although this model is simple, the proposed screen calibration method is overly complex. Zhou et al. [16] presented a personal calibration method that only requires one calibration point with a two-eye model; its estimation accuracy can reach 1.99◦. Its gaze point accuracy is relatively low, however, as the LOS is approximately presented by the line connecting the iris center and the POR. These methods estimate the POR according to the vector superposition of middle vectors, so the accuracy of the POR is very dependent on the accurate detection and estimation of each parameter. To this effect, these methods have relatively low accuracy on the whole. Unlike this, Wang and Ji [27] proposed a real-time eye gaze tracking method using a 3D deformable eye–face model. They first constructed the generic 3D deformable eye–face model based on the recovered 3D rigid facial landmarks and the estimated 3D eyeball center. The personal eye parameters and individual 3D eye–face model were then determined during a personal calibration to estimate 3D eye gaze. This method is more robust against head movement compared to other modelbased methods; however, more features are needed, such as facial landmarks, and the gaze estimation framework is more sophisticated. B. Common-Camera-Based Gaze Tracking Technology Common-camera-based gaze tracking methods usually estimate the 3D LOS based on pupil refraction and cornea reflection. This begins with determination of the user’s eye-invariant parameters. These eye-invariant parameters are then used to estimate eye variable parameters during the 3D gaze tracking process; the OA is reconstructed and the 3D LOS of the eye is estimated accordingly. Eye invariant parameters are invariant for each user but with pronounced individual differences (e.g. the cornea radius, iris radius, and kappa angle). Eye variable parameters are those which are closely related to the 3D LOS during eye movement (e.g., the cornea center, pupil center, and iris center). Common-camera-based gaze tracking systems can be divided into single-camera-single-light-source, singlecamera-multi-light-source, and multi-camera system categories based on their respective complexity. 1) Single-Camera-Single-Light-Source System: A few geometric methods do include the use of a single-camera system to estimate 3D gazes relying on population averages for certain eye parameters. Guestrin and Eizenmen [18] demonstrated that the POR can be estimated in a single-camera-single-light-source system if the distance between the eye and the computer screen is known or the user’s head is fixed, provided that the cornea radius, DCP, and indices of refraction are all known. These a priori parameters differ between different users, so they must be determined individually rather than using fixed values for all users. This is not possible with traditional single-light-source 3D gaze tracking methods. Ohno et al. [19] represented the 3D gaze by the OA based on a network camera and an infrared LED array, wherein the cornea radius, DCP, and refractive index were initially set as fixed values taken from the literature, and then Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on May 05,2021 at 12:08:13 UTC from IEEE Xplore. Restrictions apply. LIU et al.: 3D MODEL-BASED GAZE TRACKING VIA IRIS FEATURES WITH A SINGLE CAMERA AND A SINGLE LIGHT SOURCE 77 adjusted between 10% smaller and 10% larger in their experiment. They found that the gaze estimation accuracy improves when the DCP of each user is accurately determined. 2) Single-Camera-Multi-Light-Source System: 3D gaze estimation can be achieved in a single-camera-multi-light-source system [28], [33]. Villanueva and Cabeza [20] demonstrated that the minimal hardware needed for the geometric model, which is based on glint positions and pupil ellipse in the image, is a single camera and two light sources. They presented a typical such method. First, when the cornea radius is known, the cornea center can be solved as it consistently falls into the reflection planes and the distance from each reflection point on the cornea surface to the cornea center is equal; the pupil center can be solved by the refractions of pupil edge points, which satisfies the distance from each pupil edge point to the pupil center is equal. Thus, the OA can be reconstructed by the cornea center and the pupil center to determine the LOS. Representing the 3D positions of the cornea center and the pupil center by a series of nonlinear equations can reduce the calculation speed. Other researchers have used preset eye parameters to replace complex computational process at the expense of estimation accuracy. Morimoto et al.[21], for example, modeled a spherical convex mirror for cornea center estimation; they defined the gaze direction by the vector connecting the cornea center and the pupil center, wherein the cornea radius, cornea index of refraction, and the DCP data were taken from the Gullstrand model. Certain individual differences, however, were not taken into consideration. 3) Multi-Camera System: Most 3D model-based gaze estimation methods are operated in multi-camera systems [17], [18], [22]–[25], [34]. Shih and Liu [17] demonstrated that the 3D LOS computed as per the 3D position of the cornea center requires at least two cameras and at least two light sources, when the eye-specific parameters are unknown. The cornea center can be determined by the intersection of the reflection planes of multiple light sources, and then the OA can be directly denoted by the intersection line of the refraction planes as the cornea center and the pupil center both fall into the pupil refraction plane [18], [22], or reconstructed by the cornea center and the pupil center, in which the pupil center needs to be estimated by the relation of pupil refraction first [17], [23]–[25]. Guestrin and Eizenman [23] presented a POR estimation method that tolerates head movements and requires a one-point calibration procedure with three cameras and multiple light sources. They found that it would be more robust using more calibration points for user calibration. Beymer and Flickner [25] used a wide-angle stereo system to detect the face under free head movement and steered an active narrow FOV stereo system to track the eye at high resolution. 3D gaze estimation based on multiple cameras has a straightforward user calibration process and high accuracy even under natural head movement conditions, but the system calibration process is relatively complex. In summary, most 3D gaze trackers consist of a single camera and multiple light sources or multiple cameras and multiple light sources. These trackers have workable gaze estimation models and fairly accurate LOS output, but require complicated system calibration processes. Certain eye-invariant parameters (e.g., Fig. 1. Eye structure. cornea radius, DCP) must be known in single-light-source 3D gaze tracking systems to accurately estimate the 3D gaze. They are usually predefined for all users according to the population averages due to a lack of standardized equations for solving them. This drives down the estimation accuracy of these systems on the whole. Therefore, it is challenging to calibrate the eyeinvariant parameters in a single-light-source system through user calibration to achieve universal, accurate 3D gaze estimation.

### O1

Eye-tracking algorithms typically use two main approaches: feature-based and model-based approaches. Feature-based approaches detect and localize image features related to the position of the eye. A commonality among featurebased approaches is that a criteria (e.g., a threshold) is needed to decide when a feature is present or absent. The determination of an appropriate threshold is typically left as a free parameter that is adjusted by the user. The tracked features vary widely across algorithms but most often rely on intensity levels or intensity gradients. For example in infrared imagery the dual-threshold technique uses appropriately set intensity thresholds to extract the region corresponding to the pupil and the corneal reflection. The locations of the pupil and corneal reflection are then taken as the geometric center of the identified regions. The intensity gradient can also be used to detect the the pupil contour in infrared spectrum images [16,17] or the limbus in visible spectrum images [18]. An ellipse can then be fit to the feature points using least-squares fitting [19,18,16,17] or the hough transform [20]. 41 Input: Eye image, Scene image 2 Output: Point of gaze 3 Procedure: 4 Detect the corneal reflection 5 Localize the corneal reflection 6 Remove the corneal reflection 7 Iterative detection of candidate feature points 8 Apply RANSAC to find feature point consensus set 9 Determine best-fitting ellipse using consensus set 10 Model-based optimization of ellipse parameters 11 Apply calibration to estimate point of gaze Fig. 2. Starburst algorithm On the other hand, model-based approaches do not explicitly detect features but rather find the best fitting model that is consistent with the image. For example, integro-differential operators can be used to find the best-fitting circle [21] or ellipse [22] for the limbus or pupil contour. This approach requires an iterative search of the model parameter space that maximizes the integral of the derivative along the contour of the circle or ellipse. The model-based approach can provide a more precise estimate of the shape of the pupil and its center than a feature-based approach. However, model-based approaches require searching a complex parameter space that can be fraught with local minima and thus cannot be used without a good initial guess of the model parameters. The gain in accuracy of a model-based approach is obtained at a significant cost in terms of computational speed and flexibility. Notably however, the use of multi-scale image-processing methods [23] in combination with a model-based approach hold promise for real-time performance (e.g. see [15]).

# Έτοιμα συστήματα

MediaPipe Google

https://github.com/google/mediapipe

# Datasets

# Αλγόριθμοι

# Εργαλεία

### 1. OpenCV

OpenCV is an open-source machine learning and computer vision software library. Created with a view of providing a common infrastructure for [computer vision applications](https://viso.ai/applications/computer-vision-applications/), OpenCV allows access to 2,500-plus classic and state-of-the-art algorithms.

These algorithms are useful for several tasks, including [face detection](https://viso.ai/deep-learning/face-detection-overview/) and recognition, red-eye removal, object identification, extraction of 3D models of objects, [tracking moving objects](https://viso.ai/deep-learning/object-tracking/), and stitching multiple frames together into a high-resolution image.

OpenCV has multiple interfaces like C++, Python, Java, and MATLAB, and it supports most operating systems, including Windows, Android, Linux, and Mac.

**Pros:**

* Usage is free, it is open-source
* Large community support
* Offers access to more than 2,500 algorithms
* Allows you to tweak the code to serve specific purposes

**Cons:**

* It is not as easy to use as other tools like MATLAB

### 2. TensorFlow

TensorFlow is among the most popular end-to-end open-source machine learning platforms with a comprehensive set of tools, resources, and libraries. [Tensorflow](https://viso.ai/deep-learning/pytorch-vs-tensorflow/) is especially useful for building and deploying applications related to computer vision that are powered by machine learning.

TensorFlow is one of the easiest computer vision tools and allows users to develop computer vision-related machine learning models for tasks like [facial recognition](https://viso.ai/deep-learning/deep-face-recognition/), image classification, [object detection](https://viso.ai/deep-learning/object-detection/), and more. Tensorflow, like OpenCV, also supports a variety of languages like Python, C, C++, Java, or JavaScript.

**Pros:**

* It is an open-source platform
* The platform is compatible with multiple languages
* It provides constant updates for more features and improvements

**Cons:**

* It is an extremely resource-hungry toolkit

### 3. CUDA

CUDA (short for Compute Unified Device Architecture) is a parallel computing platform developed by NVIDIA. It allows developers to use the power of GPUs (Graphics Processing Units) to make processing-intensive applications faster.

The toolkit includes the NVIDIA Performance Primitives (NPP) library that provides GPU-accelerated image, video, and signal processing functions for multiple domains, including computer vision. In addition, the CUDA architecture is useful for a wide range of tasks like face recognition, image manipulation, rendition of 3D graphics, and others.

It supports various programming languages, including C, C++, Python, Fortran, or MATLAB, and is also compatible with most operating systems.

**Pros:**

* The NPP library comes with 5000-plus primitives for image and signal processing
* It includes multiple language support
* It is fast and effective

**Cons:**

* Its power consumption is quite high

### 4. YOLO

You Only Look Once, or [YOLO](https://viso.ai/deep-learning/yolov3-overview/), is among the fastest computer vision tools you can opt for in 2021. Developed by Joseph Redmon and Ali Farhadi in 2016, it was specifically made for real-time [object detection](https://viso.ai/deep-learning/object-detection/).

Faster than all other object detection tools out there, YOLO owes its speed to the application of a [neural network](https://viso.ai/deep-learning/deep-neural-network-three-popular-types/) to the complete image, which then partitions the image into grids. The software then simultaneously predicts the probabilities of each grid.

To learn more about YOLO, check out the articles [YOLOv3: Real-Time Object Detection Algorithm (What’s New?)](https://viso.ai/deep-learning/yolov3-overview/) and [YOLOv5 Is Here! Is It Real or a Fake?](https://viso.ai/deep-learning/yolov5-controversy/)

**Pros:**

* It is exceptionally fast
* The tool is highly accurate, with minimal background errors
* The algorithm has top-notch learning capabilities

**Cons:**

* It is not as effective in detecting small objects
* There is limited community support

For more in-depth information about YOLO, we suggest you read some of the other articles we’ve written discussing the nuances between the various versions of YOLO:

1. [YOLOv3: Real-Time Object Detection Algorithm (What’s New?)](https://viso.ai/deep-learning/yolov3-overview/)
2. [YOLOv5 Is Here! Is It Real or a Fake?](https://viso.ai/deep-learning/yolov5-controversy/)

### 5. MATLAB

MATLAB is a programming platform that is useful for a range of different applications such as machine learning, deep learning, and image, video, and signal processing.

It comes with a computer vision toolbox that has multiple functions, apps, and algorithms to help you design solutions for tasks related to computer vision.

**Pros:**

* It is easy to use and learn; there are many free resources on MATLAB
* Since it is a programming language, writing code is easier
* It has a convenient automatic debugging process

**Cons:**

* The tool isn’t free to use

### 6. Keras

Keras is a python-based open-source software library that acts as an interface for the machine learning platform TensorFlow. It is especially suited for beginners as it allows one to build a [neural network model](https://viso.ai/deep-learning/deep-neural-network-three-popular-types/) quickly while providing backend support.

**Pros:**

* It is user-friendly and fast
* Provides multiple backend support
* It comes with great community support

**Cons:**

* Features can be improved
* Debugging can be somewhat difficult

### 7. SimpleCV

SimpleCV is an open-source collection of libraries and software that allows you to develop computer vision applications easily. Through its framework, you gain access to several high-powered computer vision libraries such as OpenCV without the need of possessing in-depth knowledge about complex concepts like bit depths, color spaces, buffer management, or file formats.

SimpleCV is written in Python and is compatible with multiple operating systems such as Mac, Windows, and Linux.

**Pros:**

* It is free to use
* Most of the algorithms are optimized to a great extent
* Involves good documentation

**Cons:**

* It does not support any programming languages except Python

### 8. BoofCV

BoofCV is a Java-based computer vision software that is specially written for real-time computer vision solutions. It is open-source and is released under an Apache 2.0 license that makes it free to use for academic and commercial purposes.

It is a complete library with all the basic and advanced features that one may require to develop a computer vision application.

**Pros:**

* It has a user-friendly interface
* Provides multiple language support

**Cons:**

* Is slower in low-level operations

### 9. CAFFE

CAFFE or Convolutional Architecture for Fast Feature Embedding is a deep learning and computer vision framework developed at the University of California, Berkeley.

Written in C++ programming language, this framework supports multiple deep learning architectures related to image classification and segmentation. It is especially useful for research purposes and industrial implementation due to its excellent speed and image processing capabilities.

**Pros:**

* It is open-source
* Fast and easy to use
* Supports multiple languages

**Cons:**

* The documentation could be enhanced
* Provides only partial support for multi-GPU training

### 10. OpenVINO

[OpenVINO](https://viso.ai/computer-vision/intel-openvino-toolkit-overview/) (Open Visual Inference and Neural Network Optimization) is a set of comprehensive computer vision tools that are useful for developing applications emulating human vision. Developed by Intel, it is a free-to-use cross-platform toolkit.

The OpenVINO toolkit comes with models for several tasks like [object detection](https://viso.ai/deep-learning/object-detection/), face recognition, colorization, movement recognition, and more. To learn more about this tool, I recommend you to read the article [What is OpenVINO? The Ultimate Overview](https://viso.ai/computer-vision/intel-openvino-toolkit-overview/).

**Pros:**

* It is a free and efficient toolkit
* Supports multiple deep learning frameworks
* It is compatible with Windows, Mac, and Linux operating systems

**Cons:**

* Only a few examples using Python