Monserrat Aranda, C.; Rupérez Moreno, MJ.; Alcañiz Raya, ML.; Mataix, J. (2014). Markerless monocular tracking system for guided external eye surgery. Computerized Medical Imaging and Graphics. 38(8):785-792. doi:10.1016/j.compmedimag.2014.08.001.

Markerless monocular tracking system for guided external eye surgery

C. Monserrat[a,*], M. J. Rupérez[a], M. Alcañiz[a], J. Mataix[b]

[a] *LabHuman. Universitat Politècnica de València. Camino de Vera s/n, 46022, Valencia, Spain.*

[b] *Fisabio Oftalmológica Médica. Bifurcación Pío Baroja-General Avilés, S/N, 46015, Valencia, Spain.*

Abstract

This paper presents a novel markerless monocular tracking system aimed at guiding ophthalmologists during external eye surgery. This new tracking system performs a very accurate tracking of the eye by detecting invariant points using only textures that are present in the sclera, i.e., without using traditional features like the pupil and/or cornea reflections, which remain partially or totally occluded in most surgeries. Two known algorithms that compute invariant points and correspondences between pairs of images were implemented in our system: Scalable Invariant Feature Transforms (SIFT) and Speed Up Robust Features (SURF). The results of experiments performed on phantom eyes show that, with either algorithm, the developed system tracks a sphere at a 360º rotation angle with an error that is lower than 0.5%. Some experiments have also been carried out on images of real eyes showing promising behavior of the system in the presence of blood or surgical instruments during real eye surgery.

*Corresponding author

Email address: cmonserrat@labhuman.i3bh.es (C. Monserrat)

Keywords: Eye tracking, external eye surgery, markerless video tracking, SIFT, SURF and Computer Aided Surgery.

1. Introduction

The story of eye tracking begins in the last few years of the 19th century, when Louis Emile Javal [1] explained that ocular movement while reading is not continuous but is rather a sequence of short stops (fixations) and rapid movements (saccades). The first optical eye-tracking device was developed in 1901 by Dodge and Cline [2]. Some years later, Charles H. Judd [3] developed a camera that recorded ocular movement. This record permitted a detailed study of the movements of the eye using individual frame analysis. In the 1970s several researchers carried out very important studies using optical cameras [4, 5]. However, it is in the 1980s when eye tracking began to be used in Human Computer Interaction (HCI), thus marking the beginning of computer eye tracking.

The different computer eye-tracking methods that exists today can be classified into two main paradigms depending on the method used to perform the tracking: electro-oculography (EOG) [6, 7, 8, 9, 10, 11] and video-oculography (VOG). EOG computes the eye position by measuring the potential difference between the cornea and the most internal choroid layer (known as Brunch's membrane). However, this method has low accuracy and high sensitivity to external artifacts. On the other hand, VOG incorporates an optical camera to obtain images which allow the eye position/movement to be more accurately determined.

There are differences among the various VOG-based systems, which are

related to the number of cameras, the type of illumination, the observed eye structures, and the gadgets used to fix the camera position regarding the eye position. To obtain the eye position, most of the VOG systems use one (or a combination) of the following three methods:

- Purkinje images [12, 13, 14, 15, 16]: this method computes the eye position as being the difference between the first and the last Purkinje image position. The first Purkinje image is the reflection of an infra-red (IR) source from the cornea surface, and the last Purkinje image is the reflection of that same IR source from the inner layer of the crystalline.

- Dark pupil and bright pupil [17, 18, 19]: in bright pupil, the eye is illuminated with a source of IR light that is placed very near the camera axis. In dark pupil, the eye is illuminated with a source of IR light that is placed away from the camera axis.

- PCCR Vector [17], [20, 21, 22]: in this method, the vector between the pupil center and the first Purkinje image (PCCR vector) is used as the only element to compute the line of sight.

However, even though some of these systems have enough accuracy to be used for eye tracking in guided eye surgery, all of them have the same drawback: they use the pupil and/or cornea reflections to perform the eye tracking. This is a problem in guided external eye surgery since these eye components are partially or totally occluded in most of these surgeries. For this reason, a new eye-tracking approach that uses other natural features of the eye has been developed and is presented in this paper.

This work presents an economical markerless optical system that can be used as a very precise tracking system of the eye position. The aim of this eye-tracking system is to guide ophthalmologists during external eye surgery. This eye-tracking system uses natural features of the eye and allows the registration of pre-surgical images and real images of the patient's eye during surgical interventions where the pupil cannot be used as a natural tracking marker. Of all of the possible applications, two are worth mentioning:

- Surgical guidance in the treatment of the tumor of the fundus in those cases in which the pupil disappears from the field of view and the backlight pupil cannot be used in tumor localization.

- Ophthalmological guidance in the non-invasive treatment of retinal detachment by using a digital microscope, which increases the accuracy of the intervention.

To the authors' knowledge, this is the first time that a markerless optical tracking method that is not based on pupil tracking is proposed for use in external eye surgery, which is a great novelty in this field.

2. Materials and methods

In this work, a software application was developed in order to prove that optical eye tracking can be performed using a monocular system, that uses natural features of the eye, eliminating the need for especial illumination sources or external marks. For this application, the eye was modeled as a sphere with random marks which in turn modeled natural features of the eye. This artificial eye was focused on by a simple camera (Canon MV600) that

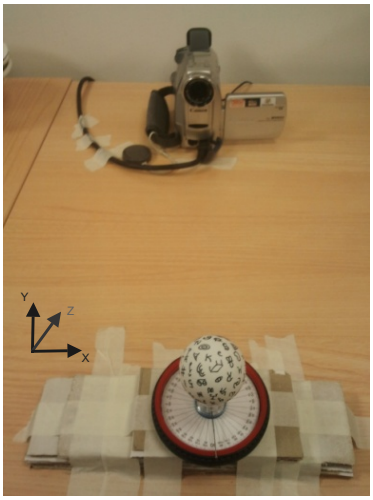

Figure 1: Hardware configuration to validate the monocular optical eye tracking. The eye is modeled as a sphere with random marks which in turn model natural features of the eye.

was connected to a PC-Computer via a FireWire digital connection. Figure 1 shows the hardware configuration used.

The developed software has the structure shown in Figure 2. The functionality of the software components can be summarized in three components: camera calibration, sphere localization, and rotation computation. The following subsections describe each software component in detail.

2.1. Camera calibration

This module calculates the intrinsic and extrinsic parameters of the camera and the radial and tangential distortion parameters. The distorsion parameters allow the aberrations caused by the camera lens to be corrected. These aberrations must be corrected because they can distort the obtained

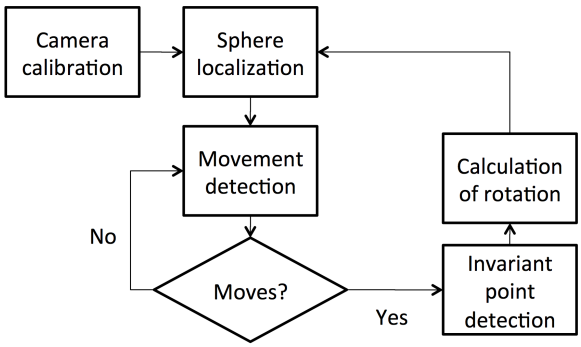

Figure 2: Software module structure. The software includes the following processes: Camera calibration (detailed in Section 2.1), Sphere localization (detailed in Section 2.2), Movement detection (detailed in Section 2.3.1), Invariant point detection (detailed in Section 2.3.2), and Calculation of rotation (detailed in Section 2.3.3).

images, which constitutes a source of error.

2.1.1. The camera model

The camera was asumed to be as a standard perspective camera. In this kind of camera, the transformation between 3D real world coordinates ($\mathbf{M} = (x, y, z)$ expressed in Euclidean space coordinates), and 2D image space coordinates ($\mathbf{m} = (u, v)$) can be modeled as shown in Eq. 1:

$$s\tilde{\mathbf{m}} = \mathbf{P}\tilde{\mathbf{M}} \tag{1}$$

where s stands for a scale factor, $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{M}}$ stand for the corresponding homogeneous coordinates of $\mathbf{m}$ and $\mathbf{M}$, and $\mathbf{P}$ stands for the projection matrix.

The projection matrix $\mathbf{P}$ can be written as shown in Eq. 2:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{2}$$

where $\mathbf{K}$ stands for the matrix of the intrinsic parameters of the camera and $[\mathbf{R}|\mathbf{t}]$ stands for the 3x4 matrix of the external parameters of the camera, which correspond to a Euclidean transformation from the world coordinate system to the camera coordinate system. $\mathbf{R}$ represents the 3x3 rotation matrix and $\mathbf{t}$ is a column vector that represents the translation.

The $\mathbf{K}$ matrix of intrinsic parameters can be written as shown in Eq. 3:

$$\mathbf{K} = \begin{bmatrix} \alpha_{\mathbf{u}} & \gamma & u_0 \\ 0 & \alpha_{\mathbf{v}} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where $\mathbf{u}$ and $\mathbf{v}$ stand for the image axes and $\alpha_{\mathbf{u}}$ and $\alpha_{\mathbf{v}}$ are the corresponding scale factors of $\mathbf{u}$ and $\mathbf{v}$. These alpha values are computed as $\alpha_{\mathbf{u}} = f \cdot m_{\mathbf{u}}$ and $\alpha_{\mathbf{v}} = f \cdot m_{\mathbf{v}}$, where $m_{\mathbf{u}}$ and $m_{\mathbf{v}}$ are scale factors that relate pixels to distance and f is the focal length in terms of distance. $\mathbf{c} = [u_0, v_0]$ stands for the intersection point of the optical axis and the image plane, and γ stands for the obliquity between the $\mathbf{u}$ and $\mathbf{v}$ image axes, which is 0 when $\mathbf{u}$ and $\mathbf{v}$ are orthogonal (which is usual in real images).

A camera is calibrated when all its internal parameters are known. However, in real cameras, the distortion parameters must also be known. The distortion can be modeled as a 2D image deformation. In this case, considering $\breve{\mathbf{u}} = [\breve{u}, \breve{v}]^T$ as the coordinates of a pixel in a distorted image and $\breve{\mathbf{x}} = [\breve{x}, \breve{y}]^T$ as its normalized coordinates, they can be related as shown in Eq. 4:

$$\breve{u} = u_0 + \alpha_v \breve{x}$$
$$\breve{v} = v_0 + \alpha_v \breve{y} \tag{4}$$

The image distortion caused by the camera lens is composed of two different distortions: radial distortion and tangential distortion. Radial distortion has a much greater effect than tangencial distortion. For this reason, most calibration algorithms ignore tangential distortion and relate the coordinates of non-distorted and distorted images as shown in Eq. 5:

$$\breve{x} = x + dx_{radial}$$
$$\breve{y} = y + dy_{radial} \tag{5}$$

where $\mathbf{x} = (x, y)$ stands for the non-distorted image coordinates, dx_{radial} stands for the radial distortion in the x axis direction, and dy_{radial} stands for the radial distortion in the y axis direction. These radial distortions can be approximated as shown in Eq. 6:

$$dx_{radial} = (1 + k_1 r^2 + k_2 r^4 + \ldots)x$$
$$dy_{radial} = (1 + k_1 r^2 + k_2 r^4 + \ldots)y \tag{6}$$

where $r = \|x\|$ and k_i stand for the coefficients of the radial distortion. In the experiments described in this work, the order used to correct the radial distorsion of the camera was 3 (only k_1 and k_2 coefficients were computed) because the values of the k_i were almost zero for coefficients of higher order.

2.1.2. Calibration Method

Using the proposed camera model, an auto-calibration photogrammetric method was programmed in order to obtain the intrinsic and extrinsic parameters of the camera. This camera calibration method is similar to the method proposed by Zhang in [23] and it was implemented using the OpenCV library. The sequence of steps needed to calibrate the camera can be summarized as follows:

1. An image pattern must be used for calibration: in our case, it was a chessboard.
2. Pictures of the pattern must be taken by the camera in different positions and orientations.
3. Some characteristic points of the image (in our case, the corners of the chessboard) must be detected in the images captured in step 2.
4. The intrinsic and extrinsic parameters of the camera can be calculated from these characteristic points.
5. These parameters can be used to obtain the radial distortion.
6. Finally, all values must be refined by minimization.

2.2. Sphere Localization

To localize the sphere in an image captured by the camera, the developed software executes three sequential steps:

1. A pre-processing step: this process transforms the input image I into an output image I' where it is easier to localize the sphere.
2. A localizing step: in this step, the software tries to localize the sphere in the pre-processed image.

3. A post-processing step: this process approximates the computation of the center and the radius of the real 3D sphere taking into account the perspective distortion from the sphere image.

The following subsections describe these steps in more detail.

2.2.1. Pre-processing

The developed algorithm uses the color information contained in the input image I for pre-processing. To perform this pre-processing efficiently, the input image color is expressed in HSV (Hue, Saturation, and Value) space color. The robustness of this pre-processing is increased by using 20 consecutive frames of the sphere images. This makes the pre-processing more independent of the presence of objects with a color similar to the one of the sphere and of variations in the illumination.

In this step, each one of the 20 frames is transformed by the following sequence of processes:

- *Transformation to the HSV color space.* The cameras usually take images in RGB color space (decomposing the light spectrum in the Red, Green and Blue channels). In this phase, the input images are transformed from RGB space to HSV space where it is easier to process them.

- *Extraction of the Value channel.* The V channel contains the information about the brightness of the image. Therefore, it is the best channel to detect the presence of the white sphere. This is the basis for processing steps that follow.

- *Simple process of image pixels.* In this step, the difference between the sphere and the background pixels is maximized. To do this, the image obtained in this step I_r is three times the difference between the V channel of the image and the inverse of this V channel.

- *Thresholding.* In order to isolate the pixels belonging to the sphere, a threshold ε is applied to the I_r image. The result is a binary image I_b where:

$$I_b(u,v) = \begin{cases} 1 & if I_r(u,v) > \varepsilon \\ 0 & otherwise \end{cases} \tag{7}$$

  All the pixels set to 1 in I_b potentially belong to the sphere. The threshold value ε was chosen as $2/3 \cdot (max - min)$, where max and min stand for the maximum and the minimum brightness of the image I_r that results from the previous processing.

- *Close morphological filter.* The Close morphological filter eliminates the black noise (isolated pixels set to 0) of the image I_b. The objective is to leave the sphere as clean as possible in order to facilitate the computation of the radius and the center of the sphere. The number of Close operations depends on the width of the marks present on the sphere surface. Five iterations of close filter were determined to be the ideal number to erase the black noise of the sphere.

- *Sphere border localization.* Once the black noise of the image is reduced, the border of the largest image object (the sphere) is extracted.

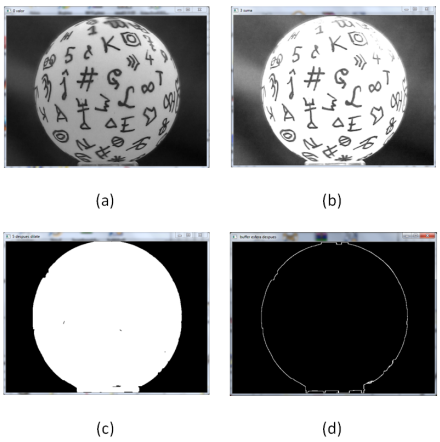

Figure 3: Images that correspond to different phases of the preprocessing step: (a) V component of the HSV color space; (b) the result of difference maximization between sphere and background pixels; (c) thresholding and morphological filter; (d) sphere border localization.

- *Insertion of the resulting image in the sphere buffer.* The algorithm accumulates the sphere edges obtained in the previous step in a single buffer called the sphere buffer. The sphere buffer is an image that has the same size as the original image I. This buffer accumulates the resulting images from the previous step.

When the preprocess ends, the algorithm has accumulated 20 sphere images in the sphere buffer. The brightest pixels in this buffer are those that can be considered the approximated sphere border. These pixels are extracted using a second threshold that results in a binary image I' with 1 for the border pixels of the sphere and 0 for the rest of the pixels. Figure 3 shows a set of images that correspond to different phases of the preprocessing step.

2.2.2. Sphere Localization

Once the border of the sphere has been approximated by the preprocessing step, the algorithm proceeds to localize the real position of the sphere. To perform this localization, the three algorithms that theoretically best meet the problem requirements were implemented to be analyzed: Hough's circles [24], Camshift [25], and the moment method [26].

To determine the best localization method, an analysis of both localization error and computational cost was performed for the three methods. The process that was used to compute the localization error compared the real center and the radius of the sphere with the computed center and the radius based on the results of the localization method. The results showed that the moment method had the lowest error and the lowest standard deviation. In addition, the moment method was almost two times faster than the other two methods. As a consequence, the moment method was the method selected for the sphere localization.

2.2.3. Refining Parameters

Once the sphere is localized in the 2D image, a sphere retro-projection must be performed in order to obtain the 3D sphere reconstruction. To do this, the perspective deformation induced by the camera model must be corrected. The perspective deformation increases with the proximity of the sphere to the camera and the deviation of the sphere center from the image center. The effect of perspective deformation is an elliptic view of the sphere in spite of its circular appearance. This ellipse is obtained by comparing each possible ellipse (of center (Cx', Cy') and main radius (R_h, R_v)) with the sphere localized in previous steps. An XOr pixel by pixel operation was used

to compare the localized sphere, and an image mask was constructed from the ellipse parameters to be considered. The ellipse parameters that minimize the XOr function were the parameters selected to perform the retro-projection of the sphere.

To retro-project the sphere, the coordinates of the points localized in the extremes of the largest main radii of the ellipse must be obtained. Those coordinates are (Lx_i, Ly_i, f) and (Lx_d, Ly_d, f), where f stands for the z coordinate of the projection plane (to facilitate understanding of the process, Figure 4 shows the retro-projection process in 2D). All points are referenced with respect to a coordinate system that is centered in a camera whose x and y axes are parallel to the x and y axes of the plane and the coordinate z is perpendicular to the projection plane. The extreme points of the ellipse mentioned above are not at the same distance from the camera. To retro-project these points, we must draw them at the same distance d from the camera on the lines defined by the points (Lx_i, Ly_i, f), $(0, 0, 0)$ and (Lx_d, Ly_d, f), $(0, 0, 0)$. The retro-projected points are (LX'_i, LY'_i, LZ'_i) and (LX'_d, LY'_d, LZ'_d), and they can be used to compute the real center of the sphere (Cx', Cy', Cz') as the midpoint of the line segment connecting these retro-projected points.

Given the retro-projected sphere, the real coordinates (Px', Py', Pz') of a point on the sphere surface can be computed from the coordinates (Px, Py, f) of the projected points on the localized sphere surface in the image. These retro-projected points are the ones that are used to track the sphere rotation.

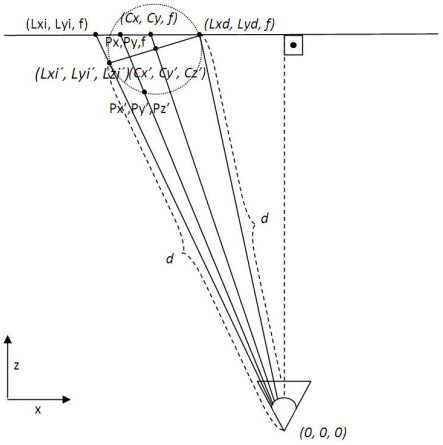

Figure 4: Sphere retro-projection in 2D. The image shows the camera (triangle), the projection plane (horizontal line), the back-projection rays (which go from the camera to the projection plane), the center of projections (the small square on the projection plane), and the sphere.

2.3. Rotation computation

Once the sphere is localized and retro-projected, the algorithm proceeds to detect its movement. In this phase, the algorithm is always testing whether or not the sphere has a significant movement. If a significant movement is detected, the algorithm takes two sets of invariant points: one set in the image before starting the movement and another set when the movement has finished. Afterwards, the algorithm obtains correspondences between the two sets of points. These correspondences allow the sphere rotation to be computed and, therefore, the sphere movement to be tracked.

2.3.1. Movement detection

To optimize the movement tracking, the movement detection process that is programmed in the algorithm only activates the invariant point detection

when the movement exceeds a certain threshold. This movement is detected by using an estimator of the optical flow that is based on the Lucas-Kanade proposal [27]. This algorithm assumes that the optical flow is constant around the pixels from the area of interest. When the flow speed vector is obtained in a sequence of images, the algorithm determines that one object (in our case, the sphere) has moved when the modulus of the speed vector exceeds a threshold. In this case, the algorithm computes the invariant points, the correspondences, and the sphere rotation based on these correspondences.

2.3.2. Computing invariant points

The techniques to obtain the invariant points from an image use local descriptors. Local descriptors search information about a localized area from an image (like textures around a point of interest) and define a codification method which is invariant to illumination, changes, and affine transformations. The implemented algorithm searches for these invariant points in two images where the object of interest has moved. Once these points are localized in both images, it obtains pairs of points that match in both images, which allow the object transformation to be computed. The most widely used current algorithms in computing invariant points and their correspondence between pairs of images are: the Scalable Invariant Feature Transforms (SIFT) [28] and the Speed Up Robust Features (SURF) [29]. In the literature, there is a large number of papers that compare SIFT and SURF, for example [30] and [31]. The following conclusions were extracted from these two works: SIFT detects more points than SURF, SIFT is a little slower than SURF, and SIFT is a little more accurate than SURF. To measure the

effectiveness of these two algorithms in eye tracking, both algorithms were implemented in our system.

2.3.3. Tracking the sphere rotation

Once the correspondences between two sets of points P_1 and P_2 are calculated (each of which is detected on the sphere surface in two consecutive images I_1 and I_2, respectively), the relative rotation performed by the sphere from image I_1 to the image I_2 can be approximated. To do this, two matrices, $\mathbf{M_1}$ and $\mathbf{M_2}$, were constructed containing the set of points P_1 and P_2, respectively, as shown in Eq. 8:

$$\mathbf{M}_1 = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}; \ \mathbf{M}_2 = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ x'_2 & y'_2 & z'_2 \\ \vdots & \vdots & \vdots \\ x'_n & y'_n & z'_n \end{bmatrix} \tag{8}$$

These matrices are ordered by point matches, in other words, the row i of matrix $\mathbf{M}_1$ stores the point coordinates contained in the set of points P_1 that matches the point coordinates contained in the set of points P_2 that is stored in the same row i of matrix $\mathbf{M}_2$. The correspondence between $\mathbf{M}_1$ and $\mathbf{M}_2$ can be written as shown in Eq. 9:

$$\mathbf{M}_2 = \mathbf{R}\mathbf{M}_1 \tag{9}$$

where $\mathbf{R}$ stands for the rotation matrix that contains the unknowns that must be obtained by solving Eq. 9. As is easily deduced, Eq. 9 is an overdetermined system of linear equations that is solved using the method proposed

by Kabsch [32]. This method, which is based on the calculation of the covariance of both matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, performs the decomposition in eigenvalues of this covariance matrix and finally computes R. To do this, at least four correspondences are necessary. However, quite a few more correspondences are recommended in order to increase the robustness of the calculations. In the experiments shown in this paper, we observed that 10 was the minimum number of correspondences that significantly decreased the error, and, therefore, it is the minimum number of correspondences required to be able to compute the rotation matrix.

3. Results

The accuracy of the implemented tracking methods using both algorithms, SIFT and SURF, were analyzed by performing a set of experiments which consisted of tracking 15 spheres of 20 mm of radius in rotations about the Y axis. These rotations were performed assuming an orthogonal reference system centered in the sphere, the XZ plane parallel to the table that supports the system, and the Y axis perpendicular to the table plane (see Figure 1). The computer that was used to perform all the tests presented in this paper was a PC DELL with Intel®Core™i7 860 of 2.80 GHz, NVIDIA GeForce 310 graphic card of 512 MB, and Windows™7 Operating System.

To measure the accuracy, the tracking methods were tested by tracking the spheres at a 360-degree rotation about the Y axis. In these experiments, the first captured image was used to determine whether or not the system reached the target rotation of 360 degrees. To detect this, the current image was compared with the initial image. If both images were equal, then a

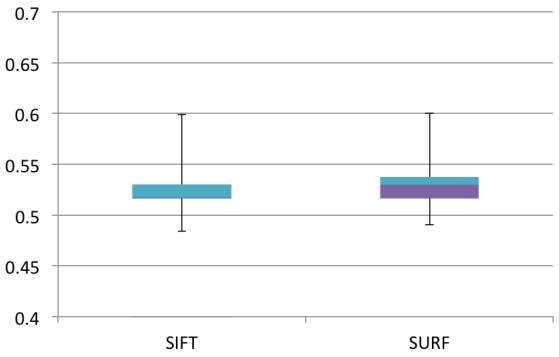

Figure 5: Error mean and standard deviation for SIFT and SURF tracking. The sphere radius used was 20 mm.

complete rotation of the sphere had been performed. Figure 5 shows the median, the 1st and 3rd quartile, and the maximum and minimum of the errors produced by each tracking method (based on SIFT or on SURF algorithm) in mm of the arc length. The accuracy of the two methods was proved since the maximum error was below 0.6 mm for both methods; however, a slightly higher error was observed for the method based on the SURF algorithm. Therefore, taking into account that the radius of the sphere was 20 mm, the committed error was under 0.5%. This means that both tracking methods had enough accuracy to be considered as tracking methods in guided external eye surgery.

To measure the effectiveness of the two methods, analyses were made of the ability of the two tracking methods to localize invariant points and to establish correspondences between points on the sphere surface in two temporal consecutive images. The results of these analyses are shown in Table 1, which shows the mean of localized descriptors, the mean temporal

cost in localizing these descriptors, the mean temporal cost in localizing correspondences between points, and the maximum angle that can be tracked between these two consecutive images.

Table 1: Effectiveness of the tracking methods.

	using SIFT	**using SURF**
Localized descriptors	966.6	1049.3
Time required to localize descriptors (s)	0.693	0.130
Time required to localize matches (s)	0.340	0.237
Maximum angle of tracked rotation (°)	27	37

The results of the analysis of the accuracy of the two tracking methods showed a slightly higher error for the method based on the SURF algorithm (see Figure 5). However, Table 1 shows that the SURF-based method was faster than the SIFT-based method. In our implementation, the SIFT-based method could compute almost 2 rotations per second compared to the 4 rotations per second that were computed by the SURF-based method. Table 1 also shows that the SURF-based method detected more correspondences than the SIFT-based method. However, the SIFT-based method detected the invariant points and their matches more accurately, which allowed the tracking method based on the SIFT algorithm to perform more accurate sphere tracking than the SURF-based method. Therefore, the SIFT-based method was the method chosen for the final implementation.

In a second experiment, the system for tracking eye movement was tested on a computer-generated eye (see Figure 6). The variable to analyze was the

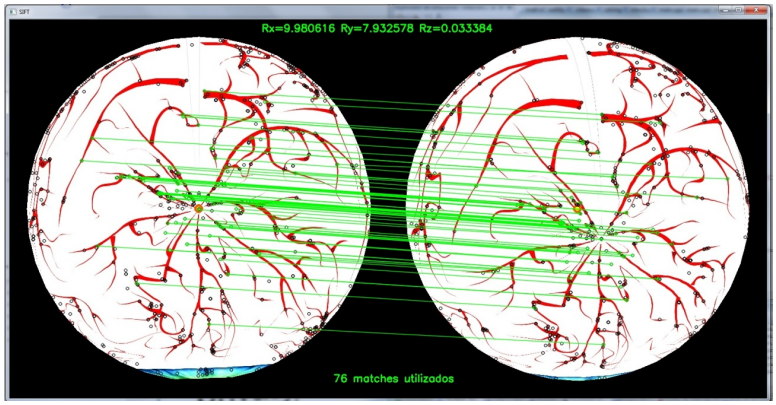

Figure 6: Example of movement tracking on a synthetic eye: the rotation was 10 degrees about the X-axis, 8 degrees about the Y-axis, and 0 degrees about the Z-axis; the detected rotation was 9.98 degrees about the X-axis, 7.93 degrees about the Y-axis, and 0.03 degrees about the Z-axis. The number of matches used was 76.

error in computation of the eye rotation angle when this eye had an unrestricted rotation axis. Table 2 shows the summary of 15 random rotations about the X-axis and Y-axis. The rotations were restricted within the interval between -27 and 27 degrees, which is the maximum rotation angle that the tracking algorithms can detect for two consecutive images (see Table 1). Table 2 shows that the system had a very low error when tracking the sphere.

Table 2: Errors in 3D synthetic eye tracking. These correspond to 15 random rotations about the X-axis and Y-axis. The errors are measured in degrees.

AXIS	X	Y	Z
Mean	0.0344	0.0279	0.0749
Variance	0.0012	0.0004	0.0073

On the other hand, in real cases, tracking the eye in external eye guided

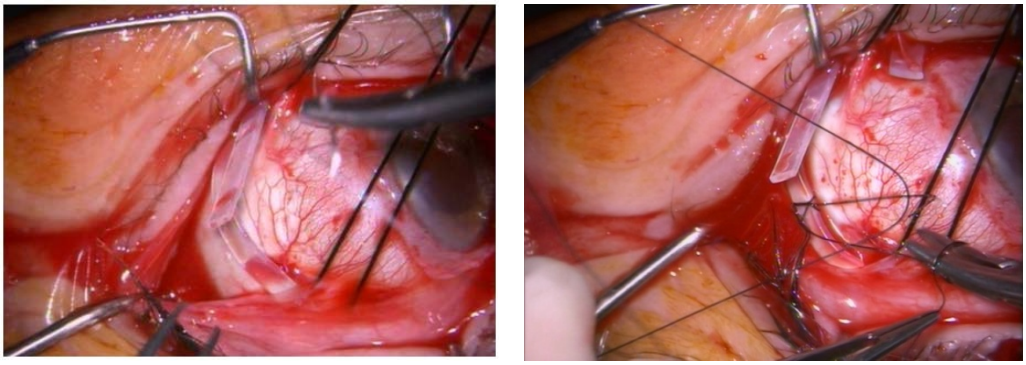

Figure 7: Successive images of eye surgery

surgery has two major problems that have to be overcome: variations in texture of the sclera surface during surgery and the presence of surgical instruments in the image of the eye. The first problem is due to the presence of blood which may appear during the dissection of the conjunctiva of the eye. This blood partially covers the sclera and masks the vascular tree of the sclera, which could hinder the location of common invariant points between successive eye images. The second problem may appear when surgery elements (like sutures, suture needle, episcleral plates, etc.) are interposed between the camera and the eye. This could partially mask the eye and it might confuse the algorithm in locating invariant points. To check the robustness of the algorithm with respect to these problems, a final experiment was designed to track real eye movement during eye surgery. Figure 7 shows different successive images during the eye surgery. As Figure 8 shows, our algorithm was capable of detecting more than 70 invariant points in each image and more than 30 correct correspondences between two consecutive eye images. This application to a real case again shows the potential of this

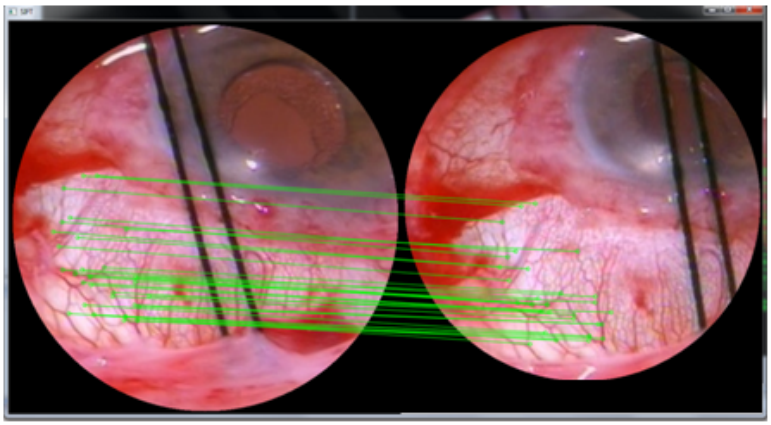

Figure 8: Invariant points and matches detected in consecutive images in the presence of surgical elements and blood.

tracking method for use in guided external eye surgery.

4. Discussion

The conclusions presented in Section 3 with regard to the SIFT and SURF analyses are similar to those obtained by previous authors (Section 2.3.2). The choice between the two algorithms will depend on what the main concern is in your problem. SIFT detects more points than SURF (i.e., it is a little more accurate), but it is a little slower than SURF. In our problem, the results have demonstrated that the error committed by the two algorithms is low enough to allow the use of either one in the tracking of eye movement. Finally, SIFT was chosen for our case because it was slightly more accurate. Nevertheless, SURF might also be used if speed is the main concern.

As mentioned at the end of Section 2, the minimum number of correspondences that should be used to obtain an acceptable error on eye phantoms

is 10. For the real case analyzed (images of eyes with blood and surgical instruments present), the number of correct correspondences was 30 (three times higher than the minimum needed). Even though it is true that the methodology still requires thorough testing on real cases, these initial results are very promising and encourage us to study the application of the tracking method on real cases.

Furthermore, the fact that the tracking system is based on a monocular optical system allows direct application to the surgical environment without adding expensive and excessive technological equipment. It is only necessary to connect the surgeon's digital microscope to a computer that has the developed tracking software installed. However, despite the potential applicability of the developed algorithm, some improvements should be made in the future.

5. Conclusions

This paper presents a novel algorithm that is able to track any spherical markerless object. The tracking is performed using only the texture of the object surface without any special reference (like pupils in eyes). If the eye is considered to be a sphere and its sclera vascular tree is considered to be the object surface texture, the results show that the developed algorithm has enough effectiveness and accuracy to be used in guided eye surgery. In addition, the results of the experiments demonstrate that our algorithm has promising behavior even with the presence of blood on the sclera and with the interposition of surgical instruments between the eye and the camera. This highlights the potential of our algorithm for tracking movements of

the eye during real surgeries. Our ongoing research is focused on hardware configuration that will allow the use of stereo cameras in eye tracking to better localize the sphere.

Acknowledgements

The authors would like to express our gratitude to Asís Tárrega as well as to the personnel from the "Fisabio Oftalmológica Médica" Eye Center.

References

[1] E. Javal, Essai sur la physiologie de la lecture, in: Annales d'Oculistique, 1879.

[2] R. Dodge, T. S. Cline, The angle velocity of eye movements., Psychological Review 8 (2) (1901) 145.

[3] C. Judd, C. McAllister, W. Steel, General introduction to a series of studies of eye movements by means of kinetoscopic photographs, Psychol. Rev., Monograph Supplements 7 (1) (1905) 1–16.

[4] K. Rayner, Eye movements in reading and information processing., Psychological bulletin 85 (3) (1978) 618.

[5] J. Merchant, R. Morrissette, J. L. Porterfield, Remote measurement of eye direction allowing subject motion over one cubic foot of space, Biomedical Engineering, IEEE Transactions on (4) (1974) 309–317.

[6] T. Zaveri, J. Winters, M. Wankhede, I. Park, A fast and accurate method for discriminating five choices with EOG, Center of Excellence, Cleveland, USA, 2006.

[7] M. Brown, M. Marmor, E. Zrenner, M. Brigell, M. Bach, et al., Iscev standard for clinical electro-oculography (EOG) 2006, Documenta ophthalmologica 113 (3) (2006) 205–212.

[8] Z. Lv, X. Wu, M. Li, C. Zhang, Implementation of the eog-based human computer interface system, in: Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on, IEEE, 2008, pp. 2188–2191.

[9] A. Bulling, D. Roggen, G. Tröster, It's in your eyes: towards context-awareness and mobile HCI using wearable EOG goggles, in: Proceedings of the 10th international conference on Ubiquitous computing, ACM, 2008, pp. 84–93.

[10] A. B. Usakli, S. Gurkan, F. Aloise, G. Vecchiato, F. Babiloni, On the use of electrooculogram for efficient human computer interfaces, Computational Intelligence and Neuroscience 2010 (2010) 1.

[11] A. Bulling, J. A. Ward, H. Gellersen, G. Troster, Eye movement analysis for activity recognition using electrooculography, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (4) (2011) 741–753.

[12] T. Ohno, N. Mukawa, A. Yoshikawa, Freegaze: a gaze tracking system for everyday gaze interaction, in: Proceedings of the 2002 symposium on Eye tracking research & applications, ACM, 2002, pp. 125–132.

[13] D. H. Yoo, J. H. Kim, B. R. Lee, M. J. Chung, Non-contact eye gaze tracking system by mapping of corneal reflections, in: Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on, IEEE, 2002, pp. 94–99.

[14] S.-W. Shih, J. Liu, A novel approach to 3-d gaze tracking using stereo cameras, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34 (1) (2004) 234–245.

[15] Z. Zhu, Q. Ji, Novel eye gaze tracking techniques under natural head movement, Biomedical Engineering, IEEE Transactions on 54 (12) (2007) 2246–2260.

[16] K. R. Park, Robust gaze estimation for human computer interaction, in: PRICAI 2006: Trends in Artificial Intelligence, Springer, 2006, pp. 1222–1226.

[17] C. H. Morimoto, D. Koons, A. Amit, M. Flickner, S. Zhai, Keeping an eye for HCI, in: Computer Graphics and Image Processing, 1999. Proceedings. XII Brazilian Symposium on, IEEE, 1999, pp. 171–176.

[18] D. H. Yoo, M. J. Chung, Non-intrusive eye gaze estimation without knowledge of eye pose, in: Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on, IEEE, 2004, pp. 785–790.

[19] C. Yang, J. Sun, J. Liu, X. Yang, D. Wang, W. Liu, A gray difference-based pre-processing for gaze tracking, in: Signal Processing (ICSP),

2010 IEEE 10th International Conference on, IEEE, 2010, pp. 1293–1296.

[20] T. E. Hutchinson, K. P. White Jr, W. N. Martin, K. C. Reichert, L. A. Frey, Human-computer interaction using eye-gaze input, Systems, Man and Cybernetics, IEEE Transactions on 19 (6) (1989) 1527–1534.

[21] L. A. Frey, K. P. White Jr, T. Hutchison, Eye-gaze word processing, Systems, Man and Cybernetics, IEEE Transactions on 20 (4) (1990) 944–950.

[22] K. P. White Jr, T. E. Hutchinson, J. M. Carley, Spatially dynamic calibration of an eye-tracking system, Systems, Man and Cybernetics, IEEE Transactions on 23 (4) (1993) 1162–1168.

[23] Z. Zhang, A flexible new technique for camera calibration, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (11) (2000) 1330–1334.

[24] R. O. Duda, P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Communications of the ACM 15 (1) (1972) 11–15.

[25] J. G. Allen, R. Y. Xu, J. S. Jin, Object tracking using camshift algorithm and multiple quantized feature spaces, in: Proceedings of the Pan-Sydney area workshop on Visual information processing, Australian Computer Society, Inc., 2004, pp. 3–7.

[26] B. Bamieh, R. De Figueiredo, A general moment-invariants/attributed-graph method for three-dimensional object recognition from a single image, Robotics and Automation, IEEE Journal of 2 (1) (1986) 31–41.

[27] B. D. Lucas, T. Kanade, et al., An iterative image registration technique with an application to stereo vision., in: IJCAI, Vol. 81, 1981, pp. 674–679.

[28] D. G. Lowe, Object recognition from local scale-invariant features, in: Computer vision, 1999. The proceedings of the seventh IEEE international conference on, Vol. 2, Ieee, 1999, pp. 1150–1157.

[29] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: Computer Vision–ECCV 2006, Springer, 2006, pp. 404–417.

[30] L. Juan, O. Gwun, A comparison of SIFT, pca-SIFT and SURF, International Journal of Image Processing (IJIP) 3 (4) (2009) 143–152.

[31] C. Valgren, A. J. Lilienthal, Sift, surf and seasons: Long-term outdoor localization using local features., in: EMCR, 2007.

[32] W. Kabsch, A solution for the best rotation to relate two sets of vectors, Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32 (5) (1976) 922–923.