Viewing Direction Estimation Based on 3D Eyeball Construction for HRI

Michael Reale, Terry Hung, and Lijun Yin

Department of Computer Science, State University of New York at Binghamton

Abstract

Natural human-robot interaction requires leveraging viewing direction information in order to recognize, respond to, and even emulate human behavior. Knowledge of the eye gaze and point of regard gives us insight into what the subject is interested in and/or who the subject is addressing. In this paper, we present a novel eye gaze estimation approach for point-of-regard (PoG) tracking. To allow for greater head pose freedom, we introduce a new calibration approach to find the 3D eyeball location, eyeball radius, and fovea position. To estimate gaze direction, we map both the iris center and iris contour points to the eyeball sphere (creating a 3D iris disk), giving us the optical axis. We then rotate the fovea accordingly and compute our final, visual axis gaze direction. Our intention is to integrate this eye gaze approach with a dual-camera system we have developed that detects the face and eyes from a fixed, wide-angle camera and directs another active pan-tilt-zoom camera to focus in on this eye region. The final system will permit natural, non-intrusive, pose-invariant PoG estimation in distance and allow user translational freedom without resorting to infrared equipment or complex hardware setups.

Keywords **—** Eye tracking, eye gaze estimation, iris modeling.

1. Introduction

Ideally, human-robot interaction (HRI) should be as unconstrained and natural as human-to-human interaction, and a key facet of natural human interaction is the behavior of the eyes. More specifically, knowledge of the viewing direction and ultimately the area of regard offers insight into the user's intention and mental focus, and consequently this information is vital for the next generation of HRI systems [5][22][23]. For example, one can readily envision robotic teaching systems that track student attention (or lack thereof) based on eye focus, mobile surveillance units that note the points of regard of suspicious individuals, robotic tour guides and salesmen that offer information based on the object the user looks at, toys that can understand and respond to what children are focusing on, and countless other applications [22].

There has been intensive research on eye tracking and gaze estimation for the past 30 years [5]. Because of the unique reflective properties of the pupil (or rather the retina) to infrared (IR) light, it is very common to use "active" lighting to detect the eyes [2][10][11]. Similarly, the overwhelming majority of gaze estimation approaches rely on glints (the reflection of infrared light off the cornea) to construct 2D or 3D gaze models [5]. While robust to visible light conditions, these methods have issues with changes in head pose, reflections off glasses, and even decreased reflectivity of the retina from contact lenses [9]. Alternatively, eye gaze may be determined from the pupil or iris contours [19] using ellipse fitting approaches [3][15]. One can also leverage the estimated iris center directly and use its distance from some reference point (e.g., the eye corners) for gaze estimation [12][20]. Indeed, the entire eye region may be segmented into the iris, sclera (white of the eye), and the surrounding skin; the resulting regions can then be matched pixel-wise with 3D rendered eyeball models (with different parameters) [18][21][22]. However, different subjects, head pose changes, and lighting conditions could significantly diminish the quality of the segmentation [21]. As a general statement, visible light techniques are usually less accurate than methods leveraging infrared light [5].

In this work, we intend to remove the need for the specialized hardware active lighting techniques require. Therefore, to bypass the issue of the diminished accuracy of iris centers from visible light imagery while simultaneously avoiding the problems heir to the potential instability of iris contour extraction, we leverage both to determine eye gaze direction with a 3D eyeball model. We calculate the 3D eyeball centers, radii, and fovea points through a unique calibration approach. During gaze estimation, we map both the estimated iris center and the iris contour points onto the eyeball sphere. After converting these points into vectors starting at the eyeball center, we then use a least-squares approach to find a vector that is at a known angle from the contour vectors while also being approximately in line with the center vector. We then find the current position of the fovea,

978-1-4244-7030-3/10/$26.00 ©2010 IEEE

which is dependent on the rotation of the eye, compute a new iris center from our initial gaze vector, and use the vector from the fovea to this new iris center as our final gaze direction. Our approach differs from the majority of previous techniques in that we map the 2D contour pixels onto the known 3D eyeball. In contrast, most gaze estimation approaches project the 3D model into the 2D plane, changing the model parameters each time until the best fit is found [18][21][22]. As such, we believe our approach is more straightforward and potentially more efficient. Moreover, when coupled with a 3D head pose and position estimation module [7], the eyeballs may be translated and rotated with head motion, thus giving us head-pose-invariant gaze estimation. See Figure 1 for an overview of our gaze estimation system.

To our knowledge, the majority of eye detection and gaze estimation approaches (particularly visible light techniques) require a high-resolution image of the face to work robustly [5]. With a conventional single-camera system, the user must keep his or her head locked in place, the camera must be strapped to the subject's head, or physical markers must be worn by the subject [2]. To address this, some systems have opted to detect the face by first finding the eyes using infrared light [10]. Alternatively, one can use stereo cameras [2][11] or construct even more complex systems (like "smart rooms" [1]). While robust, these systems generally require substantial calibration time and/or expensive non-portable setups. Even existing, non-stereo, two-camera systems [16][23] often restrict the user to a preset location in the room.

Therefore, we propose a system that first robustly locates the face and rough eye positions from a wide-angle camera view and uses this information to guide a pan-tilt-zoom camera to focus in on the eye area of the detected face (e.g., a form of "indirect search" [17]) . We apply a modified appearance-based technique based on the work by Viola and Jones [14] for face detection, and we also perform eye detection using the topographic features of eyes [15]. The pan-tilt-zoom (PTZ) camera is calibrated with the wide-angle camera by focusing the active camera on corresponding points visible in the static camera, noting the rotational values, and linearly interpolating these values for intermediate points. The details of the developed face/eye detector and the static/active camera system can be found in [6].

The resulting system allows the user translational freedom relative to the camera, and a simple adjustment of parameters will also permit varied movement in depth freedom.

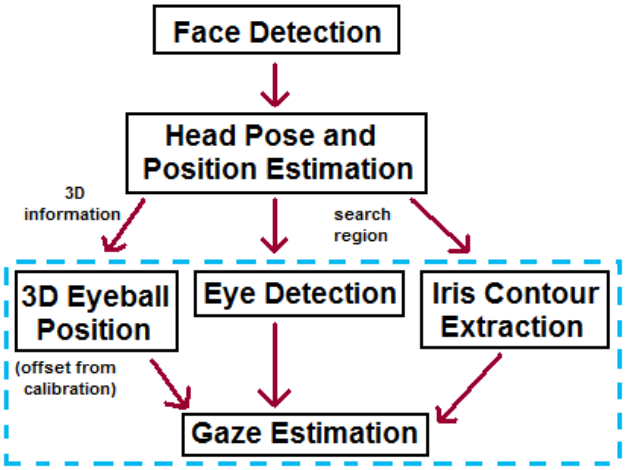

Fig. 1. Gaze estimation system overview (stages within the blue rectangle are the focus of this paper)

We intend to integrate this camera system with our gaze and PoG (Point-of-Regard) estimation approach; in this paper, however, we will focus on our eyeball model calibration and gaze estimation approaches.

In Section 2, we will describe our gaze calibration and estimation algorithm, Section 3 will present the results from our gaze estimation test, followed by a discussion and conclusion in Section 4.

2. Eye Gaze Estimation

Our eye gaze estimation model assumes that we have obtained an image with a face relatively large in frame, whether it was captured from a single, static camera near the face or from an active, optical zoom camera a distance away from the face. Once this image is acquired, we use an existing library to estimate 3D head pose and position parameters [7].

2.1. Iris detection

We work on the red channel of the image, since the iris (unlike the surrounding skin areas) generally has very low red intensity values [13]. We search for "highlight positions" on the eye (corresponding to specular reflections off the iris); these are regions containing enough dark pixels to be considered part of the iris while still having a high enough standard deviation to contain specular reflection points. To remove distracting points on glasses, we eliminate specular reflections that are long and thin (unlike iris highlights, which are generally equal in width and height). Unlike Vezhnevets and Degtiareva [13], we equalize the histogram within the eye region to increase contrast. To determine the proper thresholds and parameters, we extract information from 7 video sequences we collected using the active camera from our

two-camera system. Finally, we refine our initial iris estimate using a circular template in a very small area around the initial estimate.

2.2. Gaze calibration

Our calibration procedure has two stages: the first acquires an estimate for the 3D eyeball center and radius, and the second iteratively refines our initial estimate while also extracting the position of the fovea on the eyeball surface. Note that each eyeball is handled independently during calibration.

2.2.1. Stage 1 calibration (initial estimate)

At this point, we can get the 2D iris positions, and we assume the camera's focal length is known; given a 3D eyeball radius r, we want to find an estimate for the 3D eyeball center E.

Let us consider the camera center as the origin of a 3D coordinate system and the camera view direction as the z axis. Given a 2D pixel position and the camera's focal length, we can get a 3D vector from the camera center through every 3D position that maps to our given 2D pixel point. This can be done simply by constructing a "world image plane," a 3D plane which is perpendicular to the camera's view axis and is far away enough such that 3D points in the plane have the same x and y coordinates as their projected 2D points. A 2D pixel point can then be converted to a 3D vector by setting its z coordinate to be the z depth of the world image plane.

We operate under the assumption that, if the user is looking straight into the camera center, the eyeball center must be somewhere along a 3D vector starting at the camera center and going through the 2D iris center; that is, the gaze direction goes from the 3D eyeball center E to the 3D iris center E' (this vector is known as the "optical axis"). Granted, this is not strictly true, since the correct gaze direction (called the "visual axis") is a vector from the fovea (on the back of the eye) through the pupil [5], and this visual axis is offset from the optical axis by approximately 4-8° [4] (we will refer to this angular difference as the optical/visual axis offset). However, we will correct for this in Stage 2.

The user first looks into the camera center and then at a calibration point P whose 3D location is known. This gives us 2D iris locations $m1$ and $m2$, respectively. These points are then converted into normalized unit vectors A and B. We want to find two points, E (the eyeball center) along vector A and E' (the iris center) along vector B, such that the distance between them is r (the radius of the eyeball) and the vector between them points at P. Figure 2 illustrates this idea.

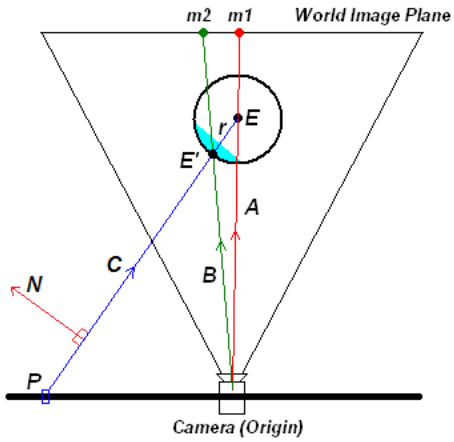

Fig. 2. Stage 1 calibration.

Let t_a and t_b represent the lengths such that $At_a = E$ and $Bt_b = E'$; we can then express our first constraint as follow:

$$\left\| At_a - Bt_b \right\| = r \tag{1}$$

Point E' (or Bt_b) is the point of intersection between B and C $(= At_a - P)$. This is equivalent to the point of intersection between B and a plane determined by point P and normal $N = (A \times B \times C)$. So, our second constraint is as follow:

$$t_b = \frac{N \bullet P}{N \bullet B} \tag{2}$$

Given above equations (1) and (2) with two unknowns t_a and t_b, we can derive the following quartic formula and solve for t_a:

$$St_a^{\,4} + Tt_a^{\,3} + Ut_a^{\,2} + Vt_a + W = 0 \tag{3}$$[1]

where:

$$S = Y^2 \tag{4}$$
$$T = -2YZ - 2(A \bullet B)XY \tag{5}$$
$$U = Z^2 + 2(A \bullet B)XZ - r^2Y^2 + X^2 \tag{6}$$
$$V = 2r^2YZ \tag{7}$$
$$W = -r^2Z^2 \tag{8}$$

and:

$$X = (A \times B \times A) \bullet P \tag{9}$$
$$Y = (A \times B \times A) \bullet B \tag{10}$$
$$Z = (A \times B \times P) \bullet B \tag{11}$$

[1] The details of the derivation of equation (3) and the solution to t_a and t_b can be found in the Section 5: Appendix A.

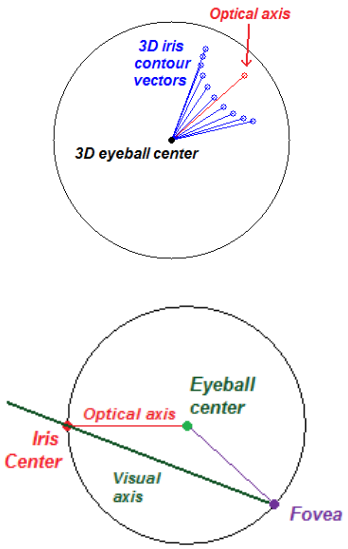

Fig. 3.a. (top) "Iris disk" concept: blue vectors are the "iris contour vectors," while the red vector is the computed (optical axis) gaze direction. **3.b. (bottom)** Optical vs. visual axis.

Once we have t_a, we can scale A by this value and get a 3D eyeball center estimate for a given radius. Although we could use a standard eyeball radius, we instead cycle through a range of 1.1cm to 1.4cm (in $1/10^{th}$ millimeter increments) and get an eyeball estimate for each radius size. This corresponds approximately to the natural range of eyeball radius sizes found in humans (1.2cm to 1.3cm) [5].

During the calibration procedure, we have the user look into the camera center first, and then look at four calibration points. The first calibration point is used to get our B vector. For each eyeball estimate (and corresponding radius) and each of the four calibration points, we use our gaze estimation approach to determine the estimated point of regard and get its distance from the true gaze (calibration) point. These error distances are added together, and the eyeball estimate (and radius) with the smallest error is chosen.

Note that this approach assumes that eyeball center (E), 3D iris center (E'), and 3D gaze point (P) are all coplanar, which may not be true if the user's head moves during the calibration procedure; thus, we must adjust our vectors for head rotation and translation before performing our calibration calculations.

2.2.2. Stage 2 calibration (refinement and fovea location)

Due to the optical/visual axis offset (as well as user error during calibration), our initial estimates can be off slightly from their true position. Therefore, we perform a secondary, iterative refinement procedure that uses a binary search to find the best eyeball center within ±5cm of our initial estimate (to a precision of $1/100^{th}$ of a millimeter). We also refine our radius search simultaneously, using the same range as before but also with $1/100^{th}$ millimeter precision.

Finally, we perform another binary search, this time moving the eyeball position up and down (within 2.5mm of the current estimate). The fovea is assumed to lie along the A vector, and we get its position by intersecting A with the sphere centered at our final eyeball center estimate.

Given the pose information which is derived from a face library [7], we have the 3D eyeball center, which can be rotated and translated with head position and pose information, the 3D eyeball radius, and the fovea point, which is stored as an offset from the eyeball center.

Note that we use 5 sample frames for each calibration point, including the look-into-camera stage.

2.3. Gaze estimation

For a given frame, we adjust the eyeball center position based on the current head pose and position. Using the 2D iris center, we want to find the iris contour points, map them (and the iris center) to the eyeball sphere, determine the optical axis, rotate the fovea point accordingly (since its location is dependent on eye position and rotation only), compute the new (3D) iris center from the optical axis, and finally get the visual axis as the vector from the fovea to the new iris center (see Figure 3.a. and 3.b. for an illustration).

To get the iris contour points, we perform the following:

1. Define a smaller search region around the iris position and blur the grayscale image with a 5x5 Gaussian filter.
2. Extract gradient magnitudes and directions.
3. Given a normalized 2D vector H that goes from our iris center estimate to the 2D eyeball center, shoot rays R_i outwards ±125° around H (one ray per degree)
4. For all pixels (within search region) along each ray, compute score:
$$score_{i,j} = m_{i,j} * (dot(H, D_{i,j}) > 0.5) \qquad (12)$$
where i is the ray index, j is the pixel index, $m_{i,j}$ is the gradient magnitude, and $D_{i,j}$ is the normalized gradient direction vector (from dark to light pixels).
5. The highest scoring pixel along each ray is chosen as a potential iris contour point, similar to [8].
6. Remove duplicate points (caused by overlapping rays), giving us N' iris contour points.
7. To eliminate possible eyelid edges, only keep pixels with gradient angles between [-35°, 35°],

[145°, 180°], and [-145°, -180°] (vertical edges only); this gives us our final N iris contour points.

8. Compute confidence value for contour as N/N'.

We map 2D iris contour points to the eyeball sphere by converting them into 3D perspective vectors and intersecting them with the current eyeball sphere. We refer to a normalized vector going from the eyeball center to one of those intersection points as an "iris contour vector" C_i. Before we can perform gaze estimation, we need to estimate the 3D iris radius on the eyeball (or rather, the average dot product between the optical axis and each C_i). We use the information extracted during the look-into-camera phase of the calibration procedure; note that an approximation of the optical gaze vector can be extracted by intersecting the 2D iris center with the eyeball sphere, subtracting the eyeball center, and normalizing the result, giving us vector V. We get the average of the dot products (*aveDot*) between V and each C_i. In this way, we define an "iris disk." Figure 3.a. illustrates this concept. This unique approach of mapping 2D contour points directly onto the 3D eyeball sphere allows for efficient detection of the gaze direction.

For gaze estimation, we first detect the iris center and contours and map these points to the eyeball sphere to ultimately get V and all C_i vectors, respectively. We want to find the optical axis G such that 1) it is parallel to V and 2) the dot product of G and each C_i is *aveDot*. Thus, we solve the following linear equations:

$$\begin{bmatrix} C_0^X & C_0^Y & C_0^Z \\ \cdots & \cdots & \cdots \\ C_N^X & C_N^Y & C_N^Z \\ V^X & V^Y & V^Z \\ \cdots & \cdots & \cdots \\ V^X & V^Y & V^Z \end{bmatrix} \begin{bmatrix} G^X \\ G^Y \\ G^Z \end{bmatrix} = \begin{bmatrix} aveDot \\ \cdots \\ aveDot \\ 1 \\ \cdots \\ 1 \end{bmatrix} \quad (13)$$

Note that *aveDot*, V, and the constant 1 are repeated in their respective matrices N times, *i.e.*, once for each contour vector.

To get the visual axis, we rotate the fovea offset based on the optical axis G. We intersect the optical axis with the eyeball sphere to get a new estimate for the 3D iris center, and take the normalized vector from the fovea to this new iris center as our final gaze direction as shown in Figure 3.b.

We have found that it is necessary to average the eyeball centers and the gaze directions over several frames to get stable estimates. For example, our current buffer for the eyeball is 10 frames, while we use 3 frames for the gaze direction. Moreover, we weigh our gaze direction by the confidence value from the contour extraction.

Once we have our current 3D eyeball position and gaze vector, we must map this information to screen coordinates. We assume that the screen's 3D position, size, and orientation are already known, and so the mapping is a simple ray-plane intersection. We use the average position of the two eyeball centers as the starting point and the average of the two visual axis vectors as our gaze direction.

For all stages of calibration and gaze estimation, we do not use a given sample of data unless its confidence value exceeds 10% (which is rather low but does at least get rid of utterly erroneous values). Moreover, while we compute calibration and gaze values for each eye separately, we do not use a sample unless both eyes have high enough confidence values.

3. Experimental Results

We performed a gaze and PoG estimation experiment wherein the user was asked to look at 20 gaze markers on the screen (effectively, the center of each brick in a 5x4 grid). As indicated before, we assume that a close-view face image captured from a distance by an active camera (using an optical zoom) is approximately equivalent to the face image captured by a static camera close the subject. As such, we use a Logitech Orbit AF web camera for this experiment. To simulate the effect of a zoom lens, the user's head was about 30 – 40 cm away from the camera on average. The image size was 640 x 480 pixels. The user focused on each marker for 2-4 seconds (about 20 gaze samples for each marker on average). Figure 4 shows an example of estimated gaze positions.

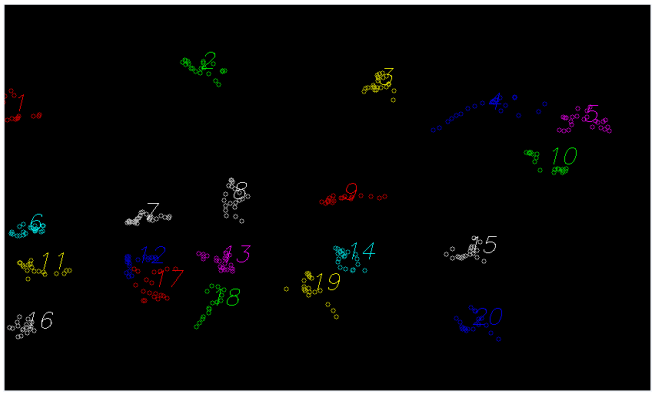

Fig. 4. Eye pointing (gaze) test. The numbers represent each cluster when the user looks at the center of blocks in positions from left to right, row by row from top to bottom.

Table 1: Intra- and inter-cluster separation (in degrees)

	Min	Max	Mean	Std Dev
Intra-cluster	0.0684°	4.9158°	1.0889°	0.6710°
Inter-cluster	2.3292°	16.4482°	8.0991°	4.7793°

Table 1 lists the variability of the gaze direction for a given gaze point ("intra-cluster") and the separation between clusters ("inter-cluster"). The latter is computed as the mean angular distance between the average gaze directions of neighbor clusters horizontal and vertically. Figure 5 shows a sample sequence of the gaze estimation results with the user's head rotated.

The ordering of the gaze points matches the pattern of the grid, and the clusters themselves are fairly compact. Our future work will involve more precisely determining the screen position relative to the camera in order to improve the eye pointing accuracy.

4. Discussion and Conclusion

In this paper, we have presented a novel eyeball center and fovea offset extraction procedure and gaze estimation approach, both of which are fully capable of dealing with head pose adjustments. When integrated with our proposed dual-camera system, we will be able to perform eye detection and gaze estimation in distance while granting the user greater translational movement freedom, without resorting to infrared light, stereo-camera setups, or other complex hardware solutions.

The integration of the gaze estimation module with our two-camera system still faces some challenges since camera rotation increases the complexity of the gaze calibration stage. For the purpose of evaluating our calibration and gaze estimation approaches, we opted to use a web camera in a special rig with a yardstick attached as a temporary arrangement. We are currently developing a mechanism to locate the screen relative to the camera, and once complete we will use points on the screen itself for calibration.

We believe the current evaluation scenario is still similar to using a zoom camera, as our active camera in the final system uses an optical zoom. Therefore, neither the resolution nor the size of facial features will be substantially different. The more pressing concern is that angular errors will result in more dramatic PoG estimation errors the farther the user is from the camera/screen. However, we do expect that the screen size increase would be approximately proportional to the distance from it.

Other future work will include dealing with larger head pose angles and increasing the speed and robustness of the iris and contour detection phases. We would also like to expand upon the secondary refinement pass to allow for a more accurate estimate of the fovea. Finally, future versions of the dual-camera system will allow for dynamic depth changes while tracking the gaze directions of multiple subjects.

Fig. 5. A sample sequence of the gaze test. Green circle is eyeball center, white circle is fovea, red circle is iris center, white lines are iris contours, and the large white lines are the gaze directions (visual axis).

5. Appendix A: Solution for t_a and t_b

Let t_a and t_b represent the lengths such that $At_a = E$ and $Bt_b = E'$, respectively. Our first constraint can be expressed as follows:

$$\left\| At_a - Bt_b \right\| = r \tag{1'}$$

We also know that the dot product of a vector with itself is its length squared, so, after multiplying out and gathering terms, we get:

$$(At_a - Bt_b) \bullet (At_a - Bt_b) = r^2 \tag{2'}$$

$$At_a \bullet At_a - At_a \bullet Bt_b - At_a \bullet Bt_b + Bt_b \bullet Bt_b = r^2 \tag{3'}$$

$$t_a{}^2 - 2t_a t_b (A \bullet B) + t_b{}^2 = r^2 \tag{4'}$$

Let us now look at our other constraint. We want to find a vector going from P to At_a such that it intersects B at Bt_b. The intersection of $C = At_a - P$ and B is equivalent to the intersection of B with a plane with point P and normal $N = (A \times B \times C)$. So:

$$t_b = \frac{N \bullet P}{N \bullet B} \tag{5'}$$

We work out N:

$$N = A \times B \times C \tag{6'}$$

$$= A \times B \times (At_a - P) \tag{7'}$$

$$= (A \times B) \times At_a - (A \times B) \times P \tag{8'}$$

$$= t_a (A \times B \times A) - (A \times B \times P) \tag{9'}$$

Substituting this for N in (5'):

$$t_b = \frac{N \bullet P}{N \bullet B} = \frac{[t_a (A \times B \times A) - (A \times B \times P)] \bullet P}{[t_a (A \times B \times A) - (A \times B \times P)] \bullet B} \tag{10'}$$

$$= \frac{t_a (A \times B \times A) \bullet P - (A \times B \times P) \bullet P}{t_a (A \times B \times A) \bullet B - (A \times B \times P) \bullet B} \tag{11'}$$

As it happens, however, $(A \times B \times P)$ is orthogonal to P, so the dot product of $(A \times B \times P)$ and P must be zero. Therefore:

$$t_b = \frac{t_a (A \times B \times A) \bullet P}{t_a (A \times B \times A) \bullet B - (A \times B \times P) \bullet B} \tag{12'}$$

For simplicity, let:

$$X = (A \times B \times A) \bullet P \tag{13'}$$

$$Y = (A \times B \times A) \bullet B \tag{14'}$$

$$Z = (A \times B \times P) \bullet B \tag{15'}$$

We can compute all of these ahead of time, so we now have t_b in terms of t_a:

$$t_b = \frac{t_a X}{t_a Y - Z} \tag{16'}$$

Substituting this into equation (4') yields:

$$t_a{}^2 - 2t_a (A \bullet B) \left[\frac{t_a X}{t_a Y - Z} \right] + \left[\frac{t_a X}{t_a Y - Z} \right]^2 = r^2 \tag{17'}$$

Multiplying by $(t_a Y - Z)^2$:

$$(t_a Y - Z)^2 t_a{}^2 - 2t_a (A \bullet B) t_a X (t_a Y - Z) + t_a X^2 = r^2 (t_a Y - Z)^2 \tag{18'}$$

We know that:

$$(t_a Y - Z)^2 = t_a{}^2 Y^2 - 2t_a YZ + Z^2 \tag{19'}$$

Substituting and multiplying everything out:

$$t_a{}^4 Y^2 - 2YZt_a{}^3 + Z^2 t_a{}^2 - 2(A \bullet B) XYt_a{}^3 + 2(A \bullet B) XZt_a{}^2 + t_a{}^2 X^2 = r^2 Y^2 t_a{}^2 - 2r^2 YZt_a + r^2 Z^2 \tag{20'}$$

Grouping like terms:

$$\begin{aligned} &(Y^2) t_a{}^4 + (-2YZ - 2(A \bullet B) XY) t_a{}^3 \\ &+ (Z^2 + 2(A \bullet B) XZ - r^2 Y^2 + X^2) t_a{}^2 \\ &+ (2r^2 YZ) t_a - r^2 Z^2 = 0 \end{aligned} \tag{21'}$$

If we create the following variables:

$$S = Y^2 \tag{22'}$$

$$T = -2YZ - 2(A \bullet B) XY \tag{23'}$$

$$U = Z^2 + 2(A \bullet B) XZ - r^2 Y^2 + X^2 \tag{24'}$$

$$V = 2r^2 YZ \tag{25'}$$

$$W = -r^2 Z^2 \tag{26'}$$

We have after substituting into (21'):

$$St_a{}^4 + Tt_a{}^3 + Ut_a{}^2 + Vt_a + W = 0 \tag{27'}$$

We can compute X, Y, Z, S, T, U, V, and W ahead of time, leaving us with a quartic formula for t_a, which is solvable. Given t_a, we can use (16') to get t_b.

6. Acknowledgements

This material is based upon the work supported by the Air Force Research Lab at Rome, NY.

7. References

[1] K. Bernardin, H. Ekenel, and R. Stiefelhagen, "Multimodal identity tracking in a smart room", *Personal and Ubiquitous Computing*, vol. 13, pp. 25-31, 2009.

[2] D. Beymer and M. Flickner, "Eye gaze tracking using an active stereo head," *CVPR'03*, vol. 2, pp. 451-458, 2003.

[3] C. Colombo, D. Comanducci, and A. Del Bimbo, "Robust tracking and remapping of eye appearance with passive computer vision," *ACM Trans. on Multimedia Computing, Communication, and Applications*, vol. 3, pp. 1-20, Dec. 2007.

[4] F. Coutinho and C. Morimoto, "Free head motion eye gaze tracking using a single camera and multiple light sources," *SIBGRAPHI'06*, pp. 171-178, Oct. 2006.

[5] D. Hansen and Q. Ji, "In the eye of the beholder: a survey of models for eyes and gaze," *IEEE Trans. on PAMI*, 32(3):478-500, 2010.

[6] Terry K. F. Hung, "Real-time face detection and applications," Master's Thesis, Binghamton University, 2008.

[7] http://www.seeingmachines.com/product/faceapi/

[8] D. Li, D. Winfield, and D. Parkhurst, "Starburst: a hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," *IEEE CVPR Workshop on Vision for Human-Computer Interaction (V4HCI)*, June 2005.

[9] K. Nguyen, C. Wagner, D. Koons, and M. Flickner, "Differences in the infrared bright pupil response of human eyes," *ACM Symposium on Eye Tracking Research and Applications (ETRA'02)*, pp. 133-138, 2002.

[10] B. Noureddin, P. Lawrence, and C. Man, "A non-contact device for tracking gaze in a human computer interface," *Computer Vision and Image Understanding*, vol. 98, pp. 52-82, 2005.

[11] T. Ohno and N. Mukawa, "A free-head, simple calibration, gaze tracking system that enables gaze-based interaction*," ACM Symposium on Eye Tracking Research and Applications (ETRA'04)*, pp. 115-122, 2004.

[12] E. Pogalin, A. Redert, I. Patras, and E. Hendriks, "Gaze tracking by using factorized likelihoods particle filtering and stereo vision," *International Symposium on 3D Data, Processing, Visualization and Transmission (3DPVT'06)*, pp. 57-64, June 2006.

[13] V. Vezhnevets and A. Degtiareva, "Robust and accurate eye contour extraction," *GraphiCon'03*, pp. 81-84, 2003.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE CVPR*, 2001.

[15] J. Wang, L. Yin, and J. Moore, "Using geometric property of topographic manifold to detect and track eyes for human computer interaction", *ACM Trans. on Multimedia Computing, Communication, and Applications*, vol.3, no. 4, pp. 1-19, Dec. 2007.

[16] J. Wang and E. Sung, "Gaze determination via images of irises," *Image and Vision Computing*, vol. 19, pp. 891-911, 2001.

[17] L.E. Wixson and D.H. Ballard, "Using intermediate objects to improve the efficiency of visual search," *Int. J. Comput. Vision 12(2*-3): 209-230, 1994.

[18] H. Wu, Y. Kitagawa, T. Wada, T. Kato, and Q. Chen, "Tracking iris contour with a 3D eye-model for gaze estimation," *ACCV'07*, LNCS 4843, pp. 688-697, 2007.

[19] D. Xia and Z. Ruan, "IR image based eye gaze estimation," *ACIS-SNPD'07*, vol. 1, pp. 220-224, 2007.

[20] J. Xie and X. Lin, "Gaze direction estimation based on natural head movements," *IEEE Inter. Conf. on Image and Graphics (ICIG'07)*, pp. 672 – 677, Aug. 2007.

[21] H. Yamazoe, A. Utsumi, T. Yonezawa, S. Abe, "Remote and head-motion-free gaze tracking for real environments with automated head-eye model calibrations," *IEEE CVPR Workshop for Human Communicative Behavior Analysis (CVPR4HB),* June, 2008.

[22] T. Yonezawa, H. Yamazoe, A. Utsumi, and S. Abe, "Gaze-communicative behavior of stuffed-toy robot with joint attention and eye contact based on ambient gaze-tracking," *Proc. of the 9th ICMI*, pp. 140-145, 2007.

[23] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, "A survey of affect recognition methods: audio, visual, and spontaneous expressions," *IEEE Trans. on PAMI, 31(1):* 39–58, 2009.

[24] S. Zhai, C. Morimoto, and S. Ihde, "Manual and gaze input cascaded (MAGIC) pointing," *ACM SIGCHI'99*, pp. 246-253, 1999.