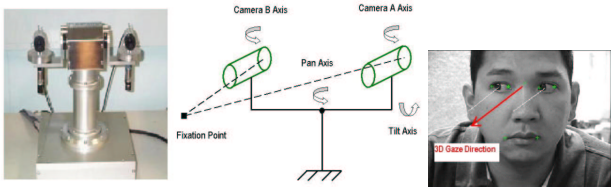
Active Gaze Tracking for Human-Robot Interaction

Rowel Atienza and Alexander Zelinsky
Research School of Information Sciences, The Australian National University
Canberra, ACT 0200 Australia
rowel|alex@syseng.anu.edu.au

Abstract

In our effort to make human-robot interfaces more user-friendly, we built an active gaze tracking system that can measure a person's gaze direction in real-time. Gaze normally tells which object in his/her surrounding a person is interested in. Therefore, it can be used as a medium for human-robot interaction like instructing a robot arm to pick a certain object a user is looking at. In this paper, we discuss how we developed and put together algorithms for zoom camera calibration, low-level control of active head, face and gaze tracking to create an active gaze tracking system.

Keywords : *Active gaze tracking, active face tracking, human-robot interface*



(a) Active vision & active head axes (b) Gaze direction

Figure 1. Active vision and gaze

Introduction

The problem with current human-robot interfaces is they are very difficult to use. For example, to program a simple pick and place task, the user has to use a teach pendant or a programming interface to tell the robot what to pick and where to put. This simple task can easily become a programming nightmare if we further assume that the object to pick can be located anywhere in the robot's workplace and can be any shape or color. This big gap in communicating what a user wants the machine to do is one of the major reasons why robots are very hard to use in our daily tasks. In our effort to make human-robot interfaces easy to use, we built an active gaze tracking system. Gaze direction normally indicates a person's interest in his/her surrounding. Therefore, it can be exploited as a very easy way to tell the robot what a user wants. For example, given the simple pick and place task mentioned earlier, a user can use gaze to instruct the robot what to pick by looking steadily at the object. Gaze can also be used to tell where to put the object (e.g. at the user's palm).

In the rest of this paper, we present an active vision system (Figure 1a) that can track a person's face and gaze in real-time (Figure 1b). Since the active vision system is like a mechanical human head and it is equipped with a pair of zoom cameras, it gives users the freedom to move without losing track of the person's face or gaze. The active gaze tracking system generates gaze and/or head pose information at approximately twice the video refresh rate (~16.5msec). During times when the gaze tracking system cannot generate data due to face occlusions or rapid head movements, a skin color-based face tracking system takes over control of the active head to make sure the user's face can still be seen at all times while the gaze tracking system is trying to produce outputs to regain control. Unlike when head pose data are available, the face tracking visual servo controller relies heavily on zoom camera intrinsic parameters to fixate since only the 2D image space face location is known. This motivated us to develop a practical zoom camera calibration technique that computes intrinsic parameters as zoom and focus are changed [1].

The following sections discuss how we built and put together the components of an active gaze tracking system. Experimental results are presented but interested readers are encouraged to view our experiment videos at: http://www.syseng.anu.edu.au/rsl/rsl_demos.html under active face or gaze tracking title.

1 Zoom Camera Calibration

In order to maintain high resolution of face images at all times, our active vision system uses a pair of zoom cameras. Since our skin color-based face tracking system can only generate 2D image space face location, the visual servo controller relies heavily on zoom camera intrinsic parameters to fixate the stereo pair. The problem is unlike static camera calibration, zoom camera calibration is many times more tedious and time consuming because of the huge number of possible lens settings. In order to make zoom camera calibration more practical, we use Willson's method [11] and assign one focus setting for each zoom assuming the distance of the object to observe can be estimated [1]. Each intrinsic parameter can be modeled as:

$$p = K_0 + K_1 z + K_2 f + K_3 z^2 + K_4 f^2 + K_5 z f \quad (1)$$

where K_i = constant, p = intrinsic parameter, f = focus and z = zoom using least squares techniques. The camera parameters obtained in this method are f_x and f_y or focal lengths in pixel dimensions along x and y axes respectively, o_x and o_y or principal point coordinate also in pixel dimensions and k_1 to k_3 or radial distortion coefficients. Focal lengths and principal point are expressed in terms of x and y pixel dimensions to incorporate in the intrinsic parameters the non-square nature of image pixels in CCD cameras.

2 Active Head Visual Servoing

The active head has four joints/axes: 1) pan, 2) tilt, 3) left camera (camera A) and 4) right camera (camera B). The objective of the control system is to move all joints together smoothly such that the fixation point can be seen at the center of left and right images at any instant of time. Furthermore, the active head must always form a symmetrical configuration as much as possible to optimize the room for left and right cameras to rotate. To make the design of visual servoing simple and robust, the control architecture distributes the task into four independent controllers instead of one homogeneous system. The controller becomes simpler and more robust compared to the one used in ESCHeR [8] or TRICLOPS [4] not only because it is distributed but it also takes advantage of the known camera intrinsic parameters and it performs smooth real time tracking by simply making joint velocity proportional to joint position error. The notion of distributed control follows the belief that the overall behavior is the most important thing after all. Before discussing the four distributed controllers, it is assumed that focal lengths, f_x and f_y and principal points o_x and o_y can be determined using Eq 1. It can also be assumed that the face location in 3D space is represented by a certain fixation point. The next section on skin color-based face tracking actually maps this fixation point as (p_x, p_y) in 2D pixel coordinates.

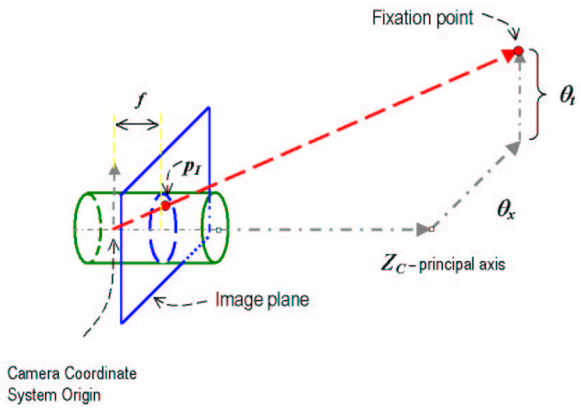


Figure 2. Fixation Point

2.1 Left and Right Cameras and Tilt Controllers

Left and right joint controllers are lumped together because their kinematics and dynamics are similar. Consider Figure 2. Here it shows the fixation point forming image point $p_I = (p_x, p_y)$. It can be easily figured out that in order to move p_I near the principal point, two subsequent rotations, θ_x and θ_t are needed. θ_x is controlled by camera axis rotation while θ_t is controlled by tilt axis rotation. θ_x can be estimated as:

$$\theta_x = \arctan\left(\frac{p_x - o_x}{f_x}\right) \quad (2)$$

The polarity of θ_x tells whether the camera should rotate clockwise or counterclockwise. A very simple yet effective controller that can make θ_x approximately zero is a P (Proportional) controller. For every video refresh cycle, the P controller calculates the necessary axis speed of rotation based on θ_x, effectively tracking the fixation point smoothly. The P controller can be summarized as:

$$v = \begin{cases} \frac{V_{max}}{\theta_x} 90 & |\theta_x| < 90\ deg \\ 0 & elsewhere \end{cases}$$

where V_{max} is a tunable parameter representing maximum safe speed of rotation for optimum response. The tilt controller follows similarly except that θ_t is used.

2.2 Pan Controller

The role of pan axis controller is to ensure that the active head physical configuration is symmetrical in order to

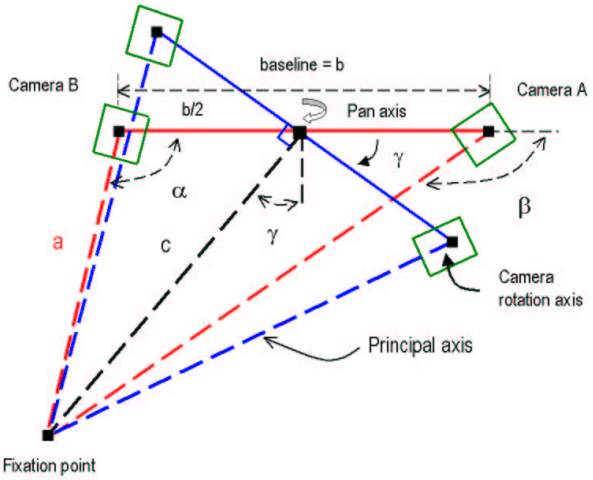


Figure 3. Top view of active head

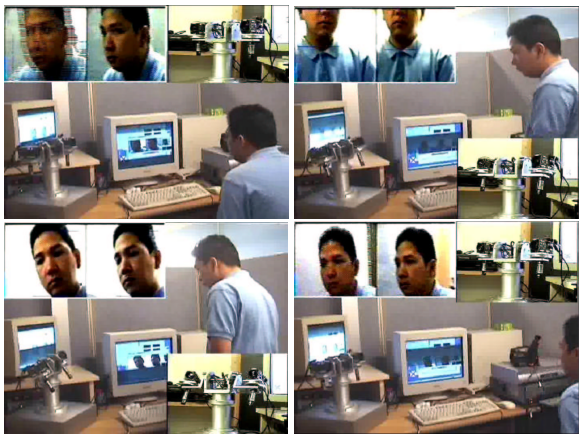


Figure 4. Active face tracking using skin color snapshots (at 5 sec interval; left-right, top-bottom). Upper left inset shows cameras A and B videos. The inset on the right is another view of the active head.

give more room for the left and right cameras to rotate in either direction. This is similar to how the human neck follows the right and left eyes tracking an object going to one side. Figure 3 shows the scenario in red triangle which is the top view of the active head. It is obvious here that in order to make the configuration symmetrical similar to the blue triangle, the pan axis must rotate by γ . The pan axis controller trusts both left and right axis controllers to follow as well because the fixation point changes in the image space when the neck rotates. To design the pan axis controller, we can use a P controller but adding an I (integral controller) helps in reducing the error quickly. The PI controller is summarized as:

$$v = \begin{cases} \frac{V_{max}}{90}\gamma + K_I \sum \gamma_{err} & |\gamma| < 90\ deg \\ 0 & elsewhere \end{cases}$$

γ can be easily computed using sine and cosine laws. The added term represents the I controller constant multiplied by the sum of the past errors.

3 Face Detection and Localization

In this section, we utilize a skin color-based face detection techniques based on skin chrominance in the CIE Lab color space [2]. The procedure is simple:

Step 1 : Build a Chrominance Chart - Several skin samples are taken from jpeg images. These samples are converted to CIE Lab's a and b components [9]. The a and b components are accumulated in an array, convolved with a 2D Gaussian Filter and normalized to form a chroma chart.

Step 2 : Use Chroma Chart for Face Detection and Localization - Color images of 320 X 240 pixels from left and right cameras are converted in CIE Lab a and b components. Since the number of pixels is huge (320 X 240 X 2), every four pixels are skipped to meet real-time requirements. The pixel's a and b components are compared with the chroma chart to determine its probability of being a skin pixel. In our experiment, if the probability is at least 0.3, it is considered as a skin pixel and labelled as such. After processing all selected pixels, the biggest skin patch is assumed to be the face. Its location (p_x, p_y) is computed as the center of the rectangle bordering the largest skin patch.

Although this technique has some limitations, (like if there is an object having the same color as skin), experimental results show that if zoom is adjusted at the same time to maintain the face image size large, real-time face tracking is possible. Some video snapshots are shown in Figure 4.

4 Active Gaze Tracking

To develop an active gaze tracking system we ported an advanced gaze tracking algorithm used in faceLAB [3] developed by Seeing Machines Inc. faceLAB generates 3D head pose (translation and rotation) and 3D gaze vector at almost twice the video refresh rate. If the true eye gaze cannot be measured due occlusions of some feature points (eye and lip corners), 3D head pose data are used to estimate gaze direction. All data are measured with respect to a fixed world coordinate (see Figure 5). Since faceLAB was designed for fixed stereo camera settings, three algorithms have to be designed before it can be used in an active vision system: 1) real-time stereo camera recalibration, 2) automatic zoom adjustment and 3) visual servo control.

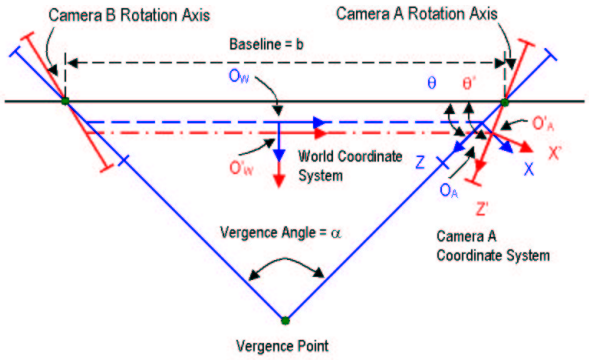


Figure 5. The stereo cameras originally in default vergence position (red line and symbols with prime) are moved to a new position (blue line). Vergence angle and vergence point are also shown.

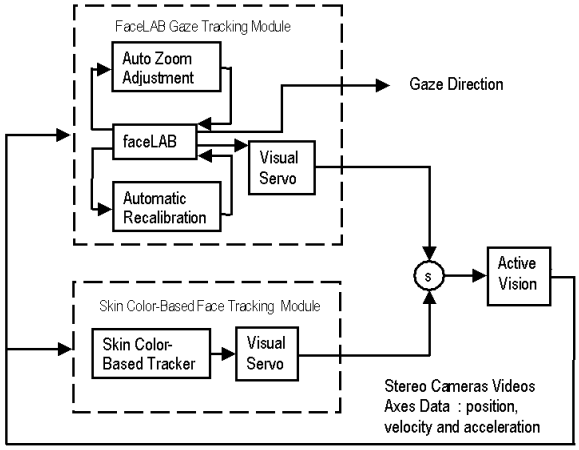


Figure 6. The software control architecture for active gaze tracking

4.1 Real-time Camera Recalibration

While tracking the user's face, the fixation point (or vergence point) of the stereo pair changes as well. Suppose the zoom setting is made fixed, a change in vergence point means the cameras are repositioned with respect to each other thus changing the extrinsic parameters. To calculate the new extrinsic parameters, consider Figure 5. Suppose cameras A and B are originally set at the default vergence angle (red line) and are fully calibrated at this position. Afterwards, they are moved to a new position (blue line). If we can determine the new extrinsic parameters at this new position, real-time recalibration is possible. Let us consider first camera A extrinsic parameters. (Computation for camera B follows similarly and no longer shown here.) The intrinsic parameters have two parts: 1) Translation of world with respect to (wrt) camera A coordinate, ${}^{A}T_W$ and 2) Rotation of world wrt to camera A, ${}^{A}R_W$. Rotation from the default vergence angle in an axis parallel to camera A Y axis changes x and z elements of ${}^{A}T_W$ only. From Figure 5 the new x and z coordinates can be easily computed as:

$$x = -\left\|{}^{A}T_W\right\| \sin(\theta) \tag{3}$$

$$z = \left\|{}^{A}T_W\right\| \cos(\theta) \tag{4}$$

where $\left\|{}^{A}T_W\right\|$ is the magnitude of the translation vector at the new vergence angle. For practical purposes, it can be assumed to be approximately equal to the known $\left\|{}^{A}T'_W\right\|$ at the default vergence angle since the distance from the camera coordinate system origin to the camera rotation point is small compared to half the stereo pair baseline.

On the other hand, rotation from the default vergence angle in an axis parallel to camera A Y axis results to a new rotation matrix given by:

$${}^{A}R_W = R_Y\left(\theta' - \theta\right){}^{A}R'_W \tag{5}$$

where ${}^{A}R'_W$ is the rotation matrix at the default vergence angle position. It must be noted that in general $y \neq 0$ and ${}^{A}R_W \neq R_Y\left(\frac{\pi}{2} - \theta\right)$ since the three axes are not perfectly aligned in the real world. Given Equations 3, 4, and 5, the new camera extrinsic parameters can now be computed while the active head is moving.

4.2 Automatic Zoom Adjustment

faceLAB has no facility to adjust camera zoom when the face image becomes too small or too big as the head distance changes. Maintaining a high face image resolution is important for effective face and gaze tracking. To rectify this problem, zoom is changed depending on the distance of the head. (Unlike in active face tracking, this case uses five zoom positions only that are calibrated by Seeing Machines' calibration tool.) Since the head distance can be obtained from the head pose data, a simple lookup table can be used to calculate the right zoom value. Focus is also adjusted to get a clear image.

4.3 Visual Servo Control System

The visual servo controller designed for face tracking can be made simpler if 3D head pose data become available. For example, in order to point the two cameras on the

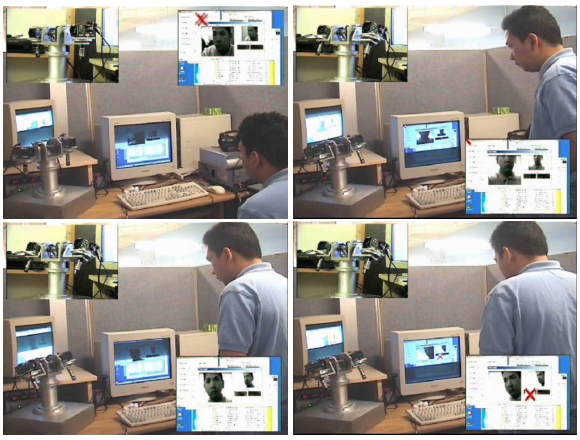


Figure 7. Both eye gaze and head pose are used to estimate gaze point (red X cursor in the PC screen inset). Snapshots are at 5 sec interval; left-right, top-bottom.

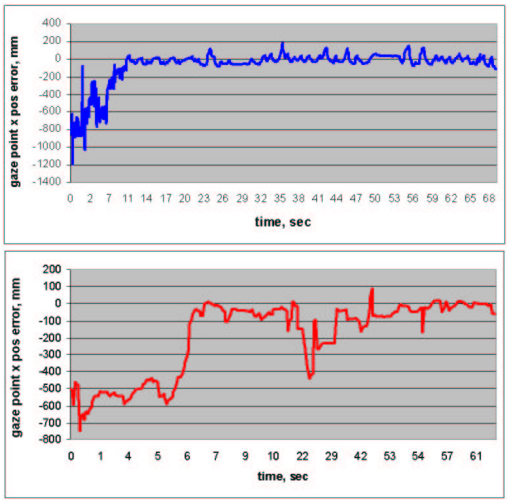


Figure 8. Typical gaze point x position error (top = using pure head pose, bottom=using pure eye gaze). Data before 11 (and 7) secs are when the user transfers his gaze from active head to screen's midpoint.

user's head, we can estimate half of the vergence angle, α, as:

$$\frac{\alpha}{2} = \arctan\left(\frac{\frac{b}{2}}{{}^{S}z_H}\right)$$

where b is the stereo camera baseline and ${}^{S}z_H$ is the translation of the person's head along the z axis of the stereo coordinate system (world coordinate rotated by tilt angle). $\frac{\alpha}{2}$ can then be used to reposition the left and right cameras using the motion control card built-in motion control algorithm.

Although the above method can also be applied to calculate the set point for tilt and pan position control, it is found that a P speed controller similar to the one used in face tracking system is more effective. A P speed controller creates a smoother transition during sudden change in set point. The only difference is instead of using 2D data, we can use the more accurate 3D head pose data expressed in stereo coordinate system. The P speed controller for tilt axis is:

$$v = K_P \arctan\left(\frac{{}^{S}y_H}{{}^{S}z_H}\right)$$

where K_P is a tunable constant and ${}^{S}y_H$ is the translation of the person's head along the y axis of the stereo coordinate system. The P speed controller for the pan axis can be computed similarly but using ${}^{S}x_H$ instead of ${}^{S}y_H$.

5 Control Architecture

In Figure 6, the control architecture integrating face and gaze tracking modules to create an active gaze tracking system is shown. The architecture bears strong resemblance to the one used in behavior-based robotics. The faceLAB module always has the priority over skin-color based tracking module in controlling the active head. Skin color-based module is only activated when faceLAB module has no output for a certain small period of time (around 60 msec).

6 Results

On Figure 7 are snapshots when both eye gaze and head pose are used to calculate gaze point while the user is trying to move his fixation point on the PC screen. Eye gaze is used when confidence is above 0.3 (out of 1.0). Otherwise, head pose is used. The gaze point on the screen can be easily calculated since the gaze vector expressed in world coordinate system is known and the rotation and translation of the screen coordinate system are known wrt to world coordinate system.

On Figure 8 is a typical gaze point x position error when: a) pure head pose or b) pure eye gaze is used to estimate gaze point while the user tries to fixate his gaze at the screen's midpoint. y position error behaves similarly.

The last experiment results in Figure 9 show a scenario when the user tries to test whether the zoom and vergence are in fact dynamically changing as he moves towards the

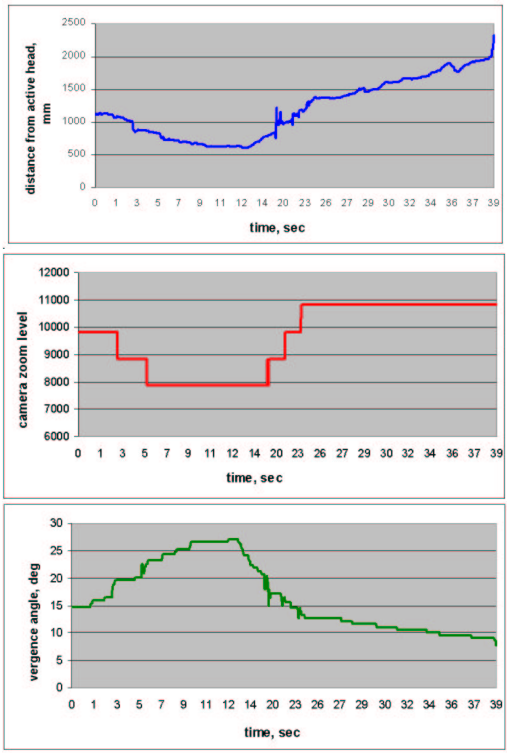


Figure 9. The user moves towards and away from the active head

active head (t = 0 to t = 13 secs) and away from it (t = 13 sec to t = 39 secs). In the top figure, it can be seen that at the current setting active gaze tracking works as far as around 2.25 meters at t = 39 sec (theoretically, it can be made to work up to 15 meters). In the middle figure, it can be seen that as the distance changes, the zoom changes correspondingly. And in the bottom figure, vergence angle also adjusts dynamically as the head distance changes.

7 Conclusion

In this paper, we presented the active gaze tracking system we developed. In contrast to using fixed camera configuration [6, 7, 12], active vision allows the user to move around the robot's workspace without losing track of the person's eye gaze. Furthermore, since gaze is measured by finding the user's irises and facial features (e.g. eye and lip corners), the eye gaze estimate is more accurate than using head pose information only [10].

Although gaze will not completely emulate a real human-human interface, we have made one aspect of human-robot means of communication more "natural". The results presented here have been promising. We intend to use this system in our future experiments on human-robot interaction.

Acknowledgement: The authors would like to acknowledge the support extended by Seeing Machines Inc., especially to Dr. Sebastien Rougeaux, on this research.

References

[1] Atienza, R. and Zelinsky, A., A Practical Zoom Camera Calibration, ACRA, Sydney, Nov. 2001.

[2] Cai, J. and Goshtasby, A., Detecting Human Faces in Color Images, Image and Vision Computing. 18: 63-75, 1999.

[3] faceLAB 1.0 User Manual, Seeing Machines Inc., 2001.

[4] Fiala, J., et. al., TRICLOPS: A Tool for Studying Active Vision, IJCV, 12:2/3, 231-250, 1994.

[5] Intel Open Computer Vision Library, http://www.intel.com/research/mrl/research/ opencv/index.htm, 2002.

[6] Matsumoto, Y. and Zelinsky, A., An Algorithm for Real-time Stereo Vision Implementation of Head Pose and Gaze Direction Measurement, Proc. of IEEE 4th International Conference on Automatic Face and Gesture Recognition, Grenoble, France, 2000.

[7] Newman, R., et. al., Real-time Stereo Tracking for Head Pose and Gaze Estimation, Proc. of IEEE 4th International Conference on Automatic Face and Gesture Recognition, Grenoble, France, 2000.

[8] Rougeaux, S., Real-Time Active Vision for Versatile Interaction, Ph. D. Dissertation, Universite d'Evry and Electrotechnical Lab, 1999.

[9] Ruzon, M., RGB to CIE Lab Matlab Code, http://vision.stanford.edu/public/ software/body.html, 1998.

[10] Stiefelhagen, R. and Yang, J., Gaze Tracking for Multimodal Human-Computer Interaction, Proc. of ICASSP, Munich, Germany, 1997.

[11] Willson, R., Modeling and Calibration of Automated Zoom Lenses, Proc. of SPIE #2350: Videometrics III, Boston MA, October 1994.

[12] Zhu, J. and Yang, J., Subpixel Eye Gaze Tracking, Proc. of the 5th International Conference on Automatic Face and Gesture Recognition, Washington, D. C., 2002.