MobiGaze: A Lightweight, Low-Cost Eye Tracking System for Mobile Brain/Body Imaging Applications

David Medine[a], Matthew Grivich[b], Scott Makeig[a]

[a] *Swartz Center for Computational Neuroscience/INC*
[b] *Neurobehavioral Systems*

1. Introduction

MobiGaze is a head-mounted eye tracker developed at the Swartz Center for Computational Neuroscience by Matthew Grivich. Eye trackers come in many flavors and are available over a wide range of prices. Since no eye tracker currently available meets the needs of certain research projects at the Swartz Center, MobiGaze was developed.

In mobile brain/body imaging experiments (MoBI <1>) the subject must be free to move about. Thus the ideal eye tracker for a MoBI situation should be portable (which suggests head mounting), comfortable and capable of determining relevant information about a subject's gaze trajectory in a 3D environment.

The first two constraints are somewhat pedestrian, but the third, is a difficult problem. Since a person may orient their gaze not only through eyeball movements, but also through head movements, a determination of head pose is necessary for valid gaze tracking in 3D.

Thus, if a subject may be free to move, conventional eye tracking systems must determine head pose as well as eyeball orientation. In the case of 'remote', or 'table-mounted', systems (where cameras and illumination are not attached to the subject), there are numerous successful methods for estimating head pose. These often use geometric approaches <2> <3> <4>. Machine learning techniques have also been proposed to deal with natural head movements <5> <6>.

These techniques are more or less effective at incorporating head movements and thereby increase the robustness of an eye tracker. But, often the goal in a remote eye tracker is merely to locate the subject's gaze in the space of a computer monitor. In a MoBI paradigm, we are often interested in a subjects' gaze at obejects or withing regions throughout the experimental laboratory.

Since MobiGaze is an eye tracker that is intended for deployment in MoBI paradigms, its natural environment is one equipped with motion capture (mocap) equipment. Thus, MobiGaze may utilize information streaming from a mocap system to locate a user's head location and regions of interest in the experimental laboratory.

In the absence of mocap technology, MobiGaze is able to determine the point of regard (POR) in a plane of the viewing frustum of a 'scene camera', which is mounted above the subject's eye to approximately 'see' what he sees. These 3D coordinates can also be projected into the 2D space of the scene camera image.

By referencing a subject's head position, we may track the movements of the scene camera since they are the same as the movements of the head itself (they are attached). Thus, the coordinates indicating the subjects gaze an be gien in the space of the mocap system. A ray between this point (the gaze vector $\vec{g}$) and the location of the scene camera (the 'eye' vector $\vec{e}$) is then extended outward to an arbitrary length. It is then a matter of simple geometry to determine if and where this ray would intersect with regions of interest in mocap space.

2. Hardware

The hardware for MobiGaze is not expensive and is fairly easy to construct. When constructed, it consists of two parts, a driver box and a modified glasses frame with cameras mounted. The glasses frame can be of any convenient manufacture, but a

Email addresses: `dmedine@ucsd.edu` (David Medine), `smakeig@ucsd.edu` (Scott Makeig)

reasonable choice is a close-fitting sunglasses frame such as the Wileyx XL-1[1]. The bottoms of the frame must be cut away so that the eye camera can view the wearer's eyeball unobstructed. It is also important that the glasses be more or less 'affixed' to the subject throughout the experiment so that shifts in position do not distort the initial calibration.

The design of the glasses is similar to previous head-mounted systems <7> <8> <9>. This is also the design used in the commercial PupilPro hardware `https://pupil-labs.com/pupil/`. In this tried and true configuration, two cameras are attached to a glasses frames. One camera – the eye camera – points toward one of the subject's eyeballs, and another – the scene camera – points outward and is positioned just above the subject's eyeball. The eye and scene cameras are mounted to the glasses frame with flexible, low-gauge copper wire so that their positions may be easily adjusted if necessary. Super glue is used for the initial attachment and a layer of epoxy is later added to strengthen the bond. The illuminating LEDs are glued to the bottom of the eye camera. Very small webcams (such as the CCIQ II camera `http://www.misumi.com.tw/PLIST.ASP?PC_ID=13`) are used so as keep the head gear as lightweight and low-power as possible. The scene camera needs to be color, but the eye camera should be black and white. Since infra-red light is used to illuminate the eye (we choose an invisible band in order to minimize distraction and discomfort) the lens must not be IR blocking.

One must also construct a driver box to hold the circuitry that powers the two cameras and two infra-red LEDs that are mounted to the glasses. The input to the driver box is a 5V DC signal. 5V batteries that use USB are common and a good choice for use as a power supply. The circuits required to provide this power is very simple and easy to construct for anyone with experience soldering. Some cable making is also necessary to construct MobiGaze. A parts list, circuit diagrams, and instructions for construction are available online: `http://sccn.ucsd.edu/labinfo/eyetracker/index.php`.

2.1. Hardware Extension: Trial Frames

We have also implemented the MobiGaze hardware setup using the 674 Simple Trial Frames manufactured by Hasegawa Bicoh Co., ltd.[2] These are adjustable glasses frames for which prescription lenses can easily be swapped in and out. The advantage is that MobiGaze can be used by subjects that require corrective lenses but for whom contact lenses are inappropriate.

[1]`https://www.wileyx.com`

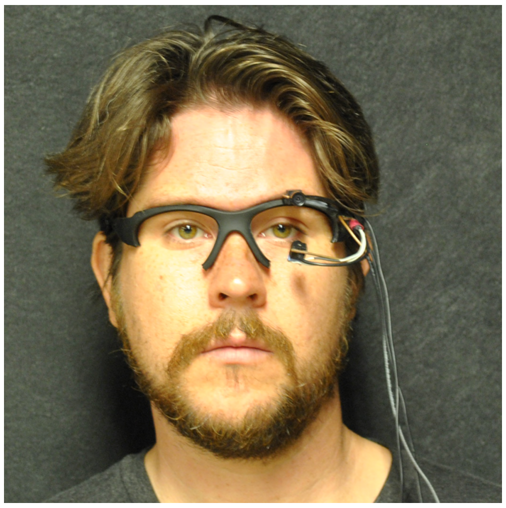


Figure 1: Wearing MobiGaze.

3. Software

The software associated with MobiGaze is (in the interest of flexibility) broken into several parts. These can be classified as either pupil tracking (estimating the pupil shape and position from the eye camera image), calibrating (determining a mapping from the estimated pupil center to points whose location are known), gaze tracking (applying this mapping to the estimated pupil position) and extensions (harvesting the gaze tracking data for interactive applications).

The constellation of MobiGaze software applications that comprise the system are written in C++ and are open source. Since these applications need to communicate data streams to one another in real time, the data streaming library Labstreaminglayer (LSL) is relied on heavily. The LSL source code host page (`https://github.com/sccn/labstreaminglayer`) also contains the source to the MobiGaze applications. Compiled versions of LSL for Windows 7 and 8 (including

[2]`http://www.bicoh.co.jp/eng_products_ophthalmic.htm`

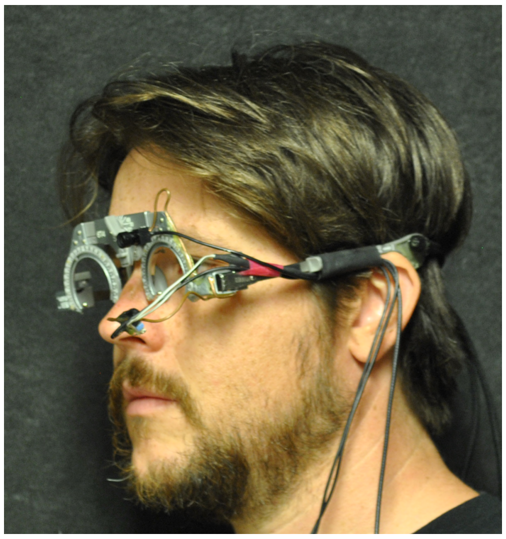


Figure 2: Wearing the trial frames version of MobiGaze

binaries for the MobiGaze applications discussed below) are available here: `ftp://sccn.ucsd.edu/pub/software/LSL/`

3.1. Pupil Tracking: Finding the Pupil Center

A most important step in any video based eye tracking application is to locate the eye from the video image. The close proximity of the eye camera to the eyeball (as is the case for most head mounted eye tracking systems) makes this relatively straight forward. Indirect IR illumination is used to create a 'dark pupil' image from which we may use computer vision techniques to derive necessary information about the pupil and corneal reflection. The physical position and focus of the eye camera hardware may be adjusted to optimize its view of the eye. Furthermore, the dimensions of the frames in which the gaze tracking algorithm searches for the pupil and corneal reflections are adjustable in the software layer.

The software application that performs the gaze tracking is called GazeStream. Once the region of interest for both pupil and corneal reflection are sufficiently sized, thresholding is applied to mark the most likely regions that contain these features. This is done by simple brightness thresholding.

The pupil is assumed to be the darkest region visible. Spatially sub-sampled points that are dark beyond the settable threshold are used as inputs to the ellipse-fitting algorithm given by <10>. The center of the fitted elise is then take as the center of the pupil.

A similar procedure is used to locate the points in the image that most likely comprise the corneal reflection. This time, a circle fitting technique <11> determines the region most likely comprises the corneal reflection. The points in this region are removed from the ellipse fitting process that estimates the pupil from the image.

In Figure 3 we see a screenshot of GazeStream in action. Here the application is in its 'Eye camera' modea and the points within the (adjustable) green box that are dark beyond a certain threshold are painted green. These are what we assume constitute the pupil and they are fed to the ellipse fitting algorithm. The center of the estimated ellipse is marked with a yellow spot by the software. This fitting may be improved by a dual-ellipse method such as the one suggested in <12>.

3.2. Calibration: Mapping the Eye Center to Gaze Points

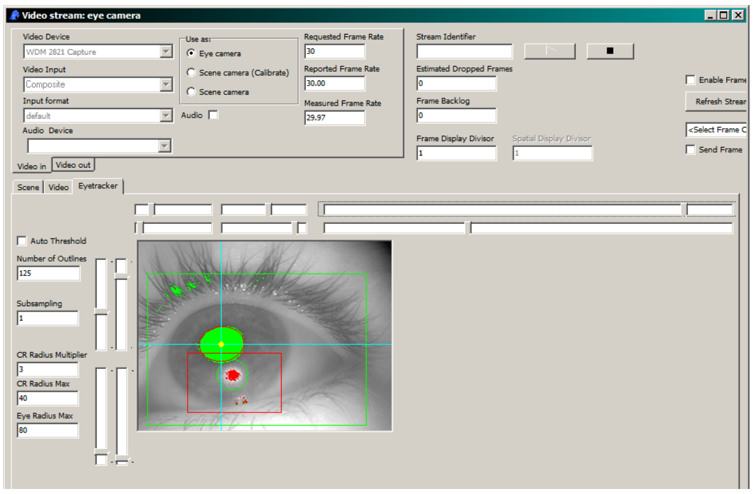
Just as GazeStream can fit an ellipse to what is most likely the pupil from the eye camera image, it can also fit any number of circles around what are most likely to be calibration spots that are displayed on a screen. Calibration patterns of colored circles are seen by the scene camera when the subject's head is oriented toward the calibration screen.

A calibration application called EyeCallbrator is responsible for displaying these calibration points and knowing their location on the calibration screen as well as the dimensions of the screen itself. EyeCalibrator also registers the center of the pupil (data passed from GazeStream in real time), and determines a mapping that projects the center of the pupil to center of the calibration targets.

Once this mapping is computed, it can be saved and summoned by various applications in the MobiGaze software constellation. One such application is GazeStream itself, which has both a 'Scene camera (Calibrate)' mode as well as a mode simply referred to as 'Scene camera' mode, which is discussed below.

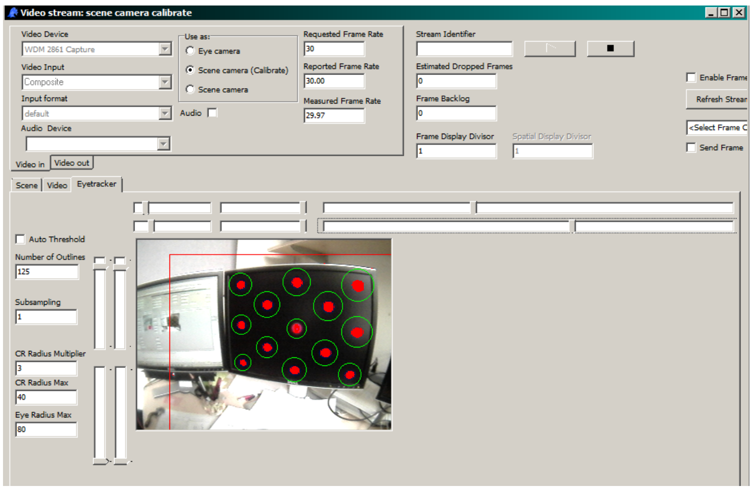
In Figure 4 we see GazeStream in its Scene camera (Calibrate) mode. The purpose of this part of the system is to pinpoint the locations of calibration targets in scene camera space. This data is then fed to EyeCalibrator. Note that since these coordinates are updated in real time, the subject is free to move her head somewhat during calibration without disturbing the accuracy.

During calibration, the subject is presented with a pattern of calibration targets such as in the one


Video stream: eye camera

Video Device: WDM 2821 Capture
Video Input: Composite
Input format: default
Audio Device:
Use as: [✓ Eye camera] Scene camera (Calibrate) Scene camera
Audio
Requested Frame Rate: 30
Reported Frame Rate: 30.00
Measured Frame Rate: 29.97
Stream Identifier:
Estimated Dropped Frames: 0
Frame Backlog: 0
Frame Display Divisor: 1
Spatial Display Divisor: 1
Enable Frame
Refresh Strear
<Select Frame C
Send Frame
Video in | Video out
Scene | Video | Eyetracker
Auto Threshold
Number of Outlines: 125
Subsampling: 1
CR Radius Multiplier: 3
CR Radius Max: 40
Eye Radius Max: 80

Figure 3: GazeStream in Eye Camera mode.


Video stream: scene camera calibrate

Video Device: WDM 2861 Capture
Video Input: Composite
Input format: default
Audio Device:
Use as: Eye camera [✓ Scene camera (Calibrate)] Scene camera
Audio
Requested Frame Rate: 30
Reported Frame Rate: 30.00
Measured Frame Rate: 29.97
Stream Identifier:
Estimated Dropped Frames: 0
Frame Backlog: 0
Frame Display Divisor: 1
Spatial Display Divisor: 1
Enable Frame
Refresh Strear
<Select Frame C
Send Frame
Video in | Video out
Scene | Video | Eyetracker
Auto Threshold
Number of Outlines: 125
Subsampling: 1
CR Radius Multiplier: 3
CR Radius Max: 40
Eye Radius Max: 80

Figure 4: GazeStream in Scene Camera (calibration) mode.

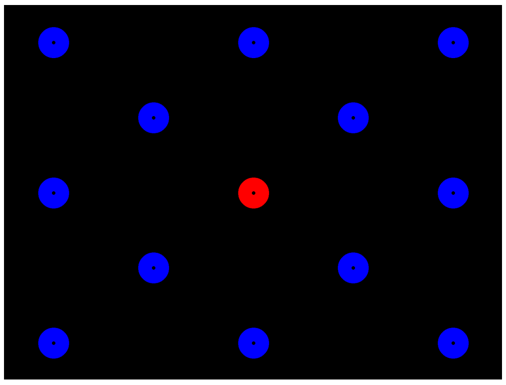


Figure 5: The calibration targets as they appear on the monitor.

shown in Figure 5. The red calibration target is the current target of interest. In the course of the calibration procedure, each target will become the sole red target. While the subject gazes on the target, his estimated pupil positions and the approximate position of the calibration target are recorded for one second. These points' mean and standard deviation are stored. This data is used to minimize a function that maps the target position to the pupil center. The inverse of this function will calculate an estimated POR from the location of the pupil center.

3.3. Calibration: Details

This section describes the mathematical and programmatic details of the MobiGaze calibration procedure. The goal of the procedure is to minimize a parametric function that transforms a calibration target in 3D 'world space' to a point that represents the center of the pupil in 2D eye camera space.

The free parameters that control our mapping function are r, the radius of the eye; $\vec{v}_0$ the vector comprised of (x_0, y_0, z_0), the distance (in mm) of the eyeball center from the scene camera; α, β, and γ, the yaw, pitch and roll of the calibration screen; $\vec{b}_{xyz}$, the 3D displacement of the eye camera from the eyeball (in mm); and l, the eye camera camera focal length (in pixels) which can (and should) be measured before hand.

The first step in the procedure is to use computer vision to help estimate the distance and pose of the scene camera relative to the calibration targets. But first, prior to estimating these parameters, the scene camera itself must be correctly calibrated. This compensates for the effects of radial and tangential distortion inherent in the camera

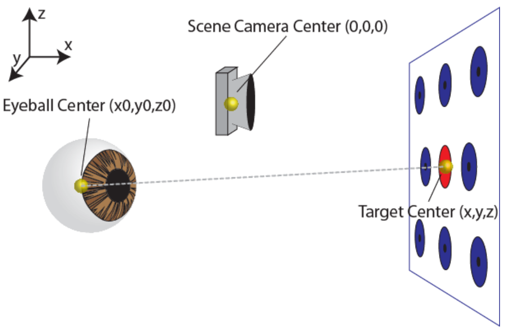


Figure 6: The geometry of the eye camera calibration procedure.

hardware. The open source computer vision library OpenCV (`http://opencv.org/`) gives a method for doing so here: `docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`.

We may lean a little further on the OpenCV library to determine the 3D coordinates of the calibration targets seen by the scene camera in a coordinate space where the camera itself is the origin. To do this, we will need to know the height and width of the calibration monitor in millimeters. EyeCalibrator uses the OpenCV function `solvePnP()` to locate each calibration target.[3] We call this point $\vec{v}$, a vector containing the values (x, y, z).

We must also save the distance of the scene camera from the target presented in the middle of the screen (which is always first in the calibration procedure). This is the value of x for that middle target. We save this as a variable d which will be used in the inverse eye mapping after the calibration parameters have been estimated.

The next step is to 'shrink-wrap' the scene camera image onto the eyeball (which we assume to be spherical). Figure 6 shows the geometry of the setup. To do this, we first subtract $\vec{v}_0$ (the eye center) from $\vec{v}$ (target location), and call this $\vec{v}_1$.

$$\vec{v_1} = \vec{v} - \vec{v_0} \tag{1}$$

We can now obtain the 'shrink-wrapped' vector $\vec{v}_2$ by normalizing $\vec{v}_1$ and multiplying by r. At this point, $\vec{v}_2$ is the calibration target of interest, projected onto a sphere of radius r. Since this vector

[3]For more details, please consult the OpenCV documentation:`http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.

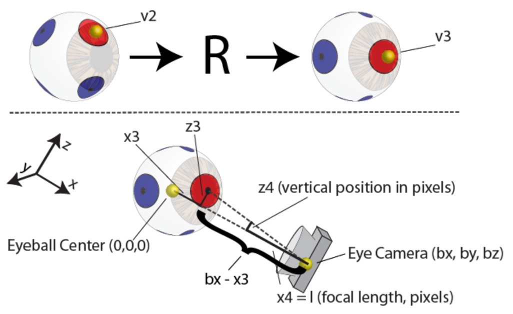

v2
R
v3
x3
z3
z4 (vertical position in pixels)
Eyeball Center (0,0,0)
Eye Camera (bx, by, bz)
bx - x3
x4 = l (focal length, pixels)

Figure 7: Above: projecting the scene camera image onto the eye and rotating it to line up with the estimated pupil position; below: extracting vertical pupil position in eye camera space from x_3 and y_3 – components of $\vec{v_3}$ – and the eye camera focal length, $x_4 = l$.

is normalized and scaled, it lies on a sphere about the origin. We are now free to consider the eye ball center to be point $(0, 0, 0)$.

$$\vec{v_2} = \frac{r\vec{v_1}}{\|\vec{v_1}\|} \tag{2}$$

We then define a rotation matrix R given by the radian values α, β, and γ.

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{3}$$

Now $\vec{v_3}$ (the calibration target projected onto the eyeball and properly rotated as shown in Figure 7) can be obtained:

$$\vec{v_3} = R\vec{v_2}. \tag{4}$$

In order to obtain the 2D coordinates of this point in the space of the eye camera (in pixels), we define a vector $\vec{v_4}$ whose components, (x_4, y_4, z_4), are:

$$\begin{gathered} x_4 = l \\ \frac{y_4}{y_3 - b_y)} = \frac{x_4}{b_x - x_3} \\ \frac{z_4}{(z_3 - b_z)} = \frac{x_4}{b_x - x_3}, \end{gathered} \tag{5}$$

where (b_x, b_y, b_z) (a.k.a $\vec{b}_{xyz}$) represents the position of the eye camera relative to the eye (which is now the origin) and $x_4 = l$ is the eye camera's focal length (in pixels). The geometric reasoning behind Equation 5 is visualized in the bottom part of part of Figure 7.

Now, the points y_4 and z_4 are estimates of where the center of the pupil should be when looking at the calibration target given by $\vec{v}$. Thus, we have all we need in order to find the values of the free parameters that will minimize the difference between (y_4, z_4) and the actual, recorded pupil centers for each calibration target.

We use the downhill simplex method <13> to find the 11 free parameters values. The algorithm used for this procedure is adapted directly from section 10.4 of ye olde *Numerical Recipes in C*<14>.

3.4. Inverting the Eye Map

Once the 11 parameters that get us from $\vec{v}$ to (y_4, z_4) are determined by the calibration procedure, we must construct an inverse eye map that gets from (y_4, z_4) to the 3D vector $\vec{v}$. To do this, we take advantage of the fact that we assume the eyeball to be a sphere.

Recall that the vector $\vec{v_3}$ is the POR shrink wrapped onto the eyeball, which is of radius r, and rotated. Thus we may state:

$$r^2 = x_3^2 + y_3^2 + z_3^2. \tag{6}$$

Furthermore, from Equation 5, we can arrive at:

$$\begin{aligned} x_3 &= bx - x_4 \\ y_3 &= \frac{y_4(b_x - x_3)}{l} + b_y \\ z_3 &= \frac{z_4(b_x - x_3)}{l} + b_z. \end{aligned} \tag{7}$$

Combining Equations 6 and 7, we arrive at a quadratic form of Equation 6, this time in terms of $(bx - x_3)$:

$$\begin{gathered} 0 = (1 + \frac{y_4^2}{l^2} + \frac{z_4^2}{l^2})(b_x - x_3)^2 + \\ (-2b_x + \frac{2y_4b_y}{l} + \frac{2z_4b_z}{l})(b_x - x_3) + \\ (b_x^2 + b_y^2 + b_z^2) - r^2. \end{gathered} \tag{8}$$

The appropriate value of $b_x - x_3$ can now be found with the quadratic formula. At this point $(b_x - x_3)$ can be substituted into 7 to give us vector $\vec{v_3}$.

The inverse of a rotation matrix is its transpose, so we can now recover $\vec{v_2}$ from $\vec{v_3}$

$$\vec{v_2} = R^T\vec{v_3}. \tag{9}$$

Since $\vec{v_2}$ is proportional to $\vec{v_1}$ we may state that

$$\begin{gathered} \frac{y - y_0}{x - x_0} = \frac{y_2}{x_2} \\ \frac{z - z_0}{x - x_0} = \frac{z_2}{x_2}. \end{gathered} \tag{10}$$

This gives us the horizontal and vertical positions of the POR at a depth of x mm. These coordinates are now all in mm. Since the information from the eye camera is 2D, without more information, we cannot estimate the true depth of the POR. Thus we set $x = d$, where d is the on-axis displacement of the eye camera from the first calibration target that we stored as part of the calibration procedure.

This is a reasonable choice because if the subject were to have his head held perfectly still by a chin strap or bite-bar, and if the PORs of interest are all displayed on the calibration monitor, $x = d$ would be absolutely true. Indeed these conditions can be created in a laboratory, but obviously, it inhibits all subject mobility. We shall see that when we co-register MobiGaze with a motion capture system, these constraints become unnecessary.

Finally, there is one caveat worth mentioning. We consider the point on axis with the center of the scene camera to have horizontal and vertical displacements of 0 mm. This means that in order to work with y_4 and z_4 we must first subtract half the height/width of the screen minus 1 pixel to center the coordinates. Without this mapping, y_4 and z_4 will be centered around the lower right-hand corner of the monitor instead of the center of the monitor.

3.5. Gaze Tracking in Scene Camera Space

It is sometimes good to pinpoint a POR in the viewing frame of the scene camera. MobiGaze is equipped with this capability as illustrated in Figure 8. In order to accomplish this, we must take into account the radial (barrel) distortion of the scene camera lens. As mentioned above, it is necessary to calibrate the scene camera prior to use. We use the returned 3 distortion coefficients, (k_1, k_2, k_3) and the calculated horizontal and vertical focal lengths (l_y, l_z) to compute a 3rd order Brown-Conrady model of the radial distortion <15>.

The horizontal and vertical and vertical displacements (y, z) that the inverse eye model returns are first scaled by the initial target distance $x = d$ which remains fixed as the depth of POR. The sum of the squares of these values is the square of the radius of the distortion model.

$$\rho = (\frac{y}{x})^2 + (\frac{z}{x})^2 \tag{11}$$

The radial distortion is then expressed as

$$\delta = 1 + k_1\rho^2 + k_2\rho^3 + k_3\rho^3; \tag{12}$$

and, finally the coordinates of the POR in the 2D space of the scene camera image are simply

$$\begin{aligned} y_s &= -l_y * \delta\tfrac{y}{x} + .5(w-1) \\ z_s &= -l_z * \delta\tfrac{z}{x} + .5(h-1) \end{aligned} \tag{13}$$

where w and h are the width and height of the scene camera image in pixels.

3.6. Extrapolating 3D Gaze Information Using Motion Capture

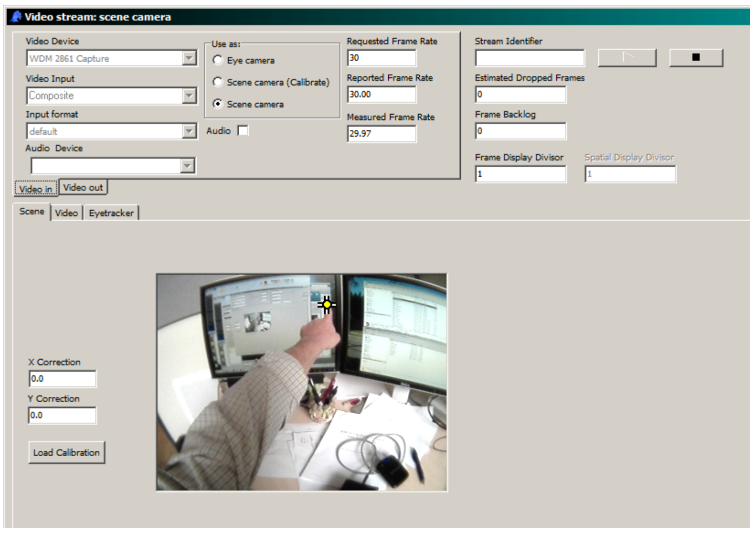
In the context of MoBI experiments, it is most useful to have information about where a subject is directing her gaze in the 3D space. Because the vector $\vec{v}$ is the mapping of the pupil position to some point that is in front of the scene camera, and because the scene camera is assumed to be the origin in this space, we can easily find a gaze vector $\vec{g}$ which is a $\vec{v}$ translated and rotated by the same amount as the location of the scene camera in mocap coordinates.

In order to accomplish this, we first must reference the initial scene camera position. This is accomplished by noting the 3D coordinates (in mocap space) of the 4 corners of the calibration screen and 4 reference markers affixed to the subject's head. Since OpenCV's pose finder method can determine the displacement of the camera relative to the screen, and since the screen's position is known, the original position of the scene camera can be referenced simultaneous to the original position of the 4 head markers.

Now the subject is free to move. Using the method of Challis to determine rigid body rotation and translations <16>, we may at any time after calibration determine an updated position of the scene camera. We consider this point (the scene camera location translated and rotated the same degree as the head markers) to be the 'eye' vector, $\vec{e}$. Given this point and the gaze vector, $\vec{g}$, we have the 2 points necessary to give an estimated gaze trajectory in motion capture coordinates.

We may then test to see where the ray $\vec{g} - \vec{e}$, when extended, intersects with the plane of the calibration screen. In order to determine the dimensions of the screen, we first must mark its corners with mocap markers. Any number of screens can be set up, but each must play a role in the calibration procedure for the software to work properly.

In theory, any robust 3D head tracking system could support registering an eye tracker in room coordinates, although currently only the PhaseSpace system is implemented. We welcome open source community development that will implement other mocap technologies (or any other improvements, for that matter). A particular advantage of the PhaseSpace system is that its active (LED emitter) markers send their marker codes to the motion


Video stream: scene camera

Video Device: WDM 2861 Capture
Video Input: Composite
Input format: default
Audio Device:

Use as:
( ) Eye camera
( ) Scene camera (Calibrate)
(•) Scene camera
Audio [ ]

Requested Frame Rate: 30
Reported Frame Rate: 30.00
Measured Frame Rate: 29.97

Stream Identifier:
Estimated Dropped Frames: 0
Frame Backlog: 0
Frame Display Divisor: 1
Spatial Display Divisor: 1

X Correction: 0.0
Y Correction: 0.0

Figure 8: GazeStream in Scene Camera mode, after calibration has been loaded.

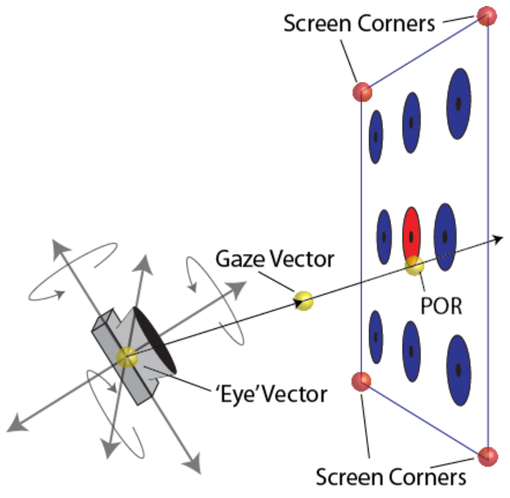


Figure 9: MobiGaze co-registered with mocap allows for robust head movement and eye tracking in 3D space.

capture cameras, eliminating the chance of confusing markers that are temporarily occluded by subject movements (or potentially, by the free movements of several subjects each wearing MobiGaze systems).

4. Applications and Accuracy

We present a variety of configurations for the MobiGaze system and analyze the accuracy for each configuration. In eye tracking, *accuracy* is defined as the angle between the ray formed by the eye and the target that the subject is supposed to be regarding and the ray formed by the eye and where the actual POR is reported <17>. The smaller this error, the higher the accuracy.

Obviously, the acuity of these measures depend on the performance of an actual subject. Thus, many manufacturers will use a mechanical eye to calibrate and test their devices. It has been suggested that this enhances the quality of eye tracking accuracy <18>. Here, we are only concerned with the accuracy of the eye tracker on human subjects as that is the only application the device is intended for.

Due to the variety of eye tracking methodologies and systems, one can only generalize about what constitutes a 'good' accuracy in human subjects. One such generalization is that accuracy within 1 degree is very difficult to achieve. Another is that one commonly sees accuracy measures in the vicin-

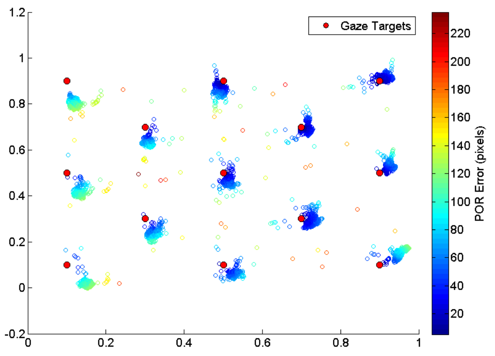


Figure 10: Targets and PORs displayed in 2D (from Protocol 1).

ity of 1.5 degrees in the literature. See for example <19> <20> <6> <21> <22> etc.

4.1. Targets on a Screen

A common application of an eye tracking system in behavioral neuroscience is to present visual stimuli to a subject on a computer monitor. In this case, we may measure accuracy simply by comparing the position of the target to the reported POR.

In order to meaningfully measure the accuracy, some data manipulation is necessary. The raw data output from MobiGaze is a duple of values. These range (when gaze is directed at a) from 0 to 1, the point 1,1 corresponding to the upper left corner of the monitor. The calibration points (given by the same coordinate system) are used as targets in the following procedures.

An example of what this looks like is given in Figure 10. Here the color corresponds to the error in pixels which we may calculate simply by multiplying by these normalized positions by the screen resolution.

If we wish to measure the angular error (accuracy) in 3D space, we must first project the target and PORs to the monitor that we are viewing. To do this, we first assume that the monitor itself is a plane. We then locate the normalized, 2D coordinates of the targets and PORs on the plane formed by 2 axes in 3D. This is done simply by adding a column of zeros to the 2D normalized coordinates. We can then project these coordinates to the plane formed by three of the corners of the display monitor.[4] This yields a picture wherein our targets and

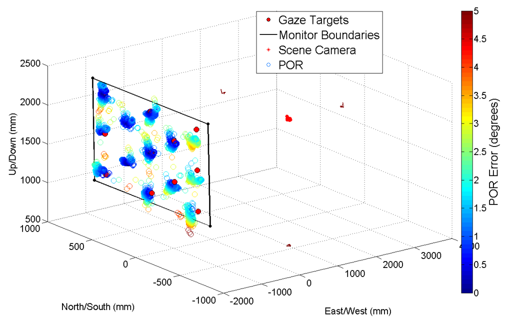


Figure 12: The data for Protocol 1.

PORs are located in 3D space as shown in Figure 12. The methods described above are used to determine the location of the scene camera at any point in time.

Protocol 1:Head Still. The mean distance from eye to target is 3.01 m in this protocol. The subject is free to move when calibrating and performing the task of gazing on each target for a period of time. By noting the actual location of the target and the location of the gaze we can calculate the angle between the vectors connecting these points to the estimated position of the scene camera for each datum. The monitor used in this experiment is a projection screen with the dimension of 1561.83 by 1057.27 mm and a resolution of 1280 by 2024 pixels.

Shown in Figure 12 is the data projected into 3D space. In Figure 11 are histograms showing the accuracy for each target point. The median accuracy overall was 1.26 degrees.

Protocol 2: Wandering Head. We may also estimate the accuracy of MobiGaze in an ambulatory situation. In this protocol, we choose 1 rather than 13 targets, but the subject is asked to wander about the room while gazing at this sole target.

The movement of the scene camera and the estimated PORs are shown in Figure 13. The horizontal and vertical range of the angle of the scene camera relative to the target is 73.64 and 43.29 degrees respectively. The range of distance traveled is 2.81 meters. The median accuracy was measured as 2.87 degrees. Figure 14 shows the accuracy (in

[4]The affine transform that performs this mapping is computed using Matlab's Procrustes method:P `http://www.mathworks.com/help/stats/procrustes.html`.

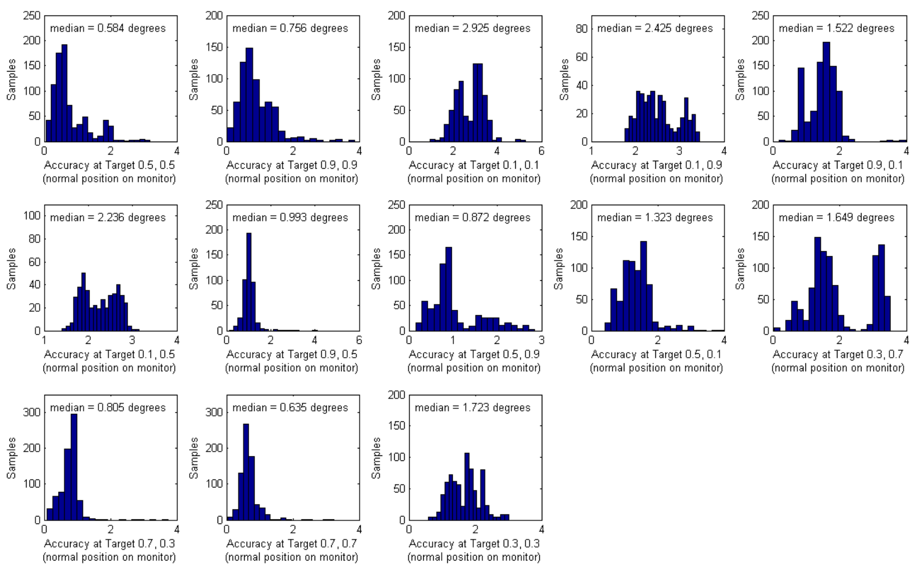


Figure 11: Histograms of accuracy at each target point for Protocol 1.

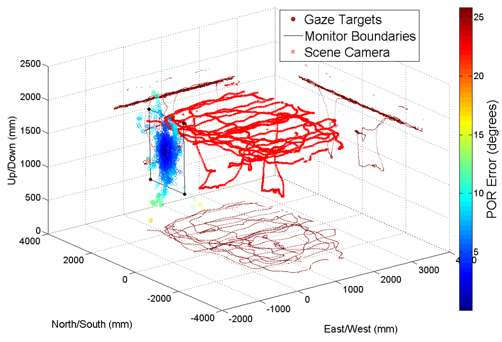


Figure 13: The subject moves freely and gazes at the center target.

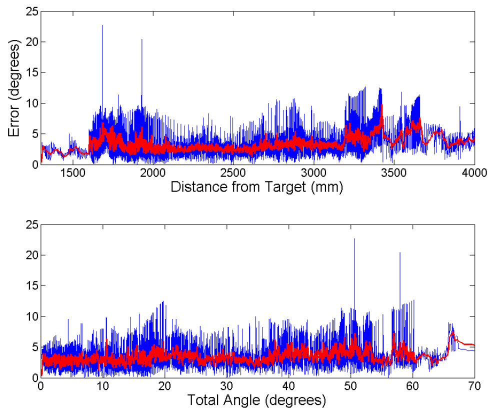


Figure 14: Accuracy as functions of distance and angle in Protocol 2.

degrees) as functions of distance and the angle between the scene camera-target vector and the vector orthogonal to the target on the screen.

Protocol 3: 3 Monitors. MobiGaze can be configured to track POR for targets that appear on multiple monitors. In this protocol we show results for such a setup. The subject sits before a triptych of monitors in portrait orientation. The monitors in this protocol are each 768 by 1366 pixels and measure 333.72 by 524.96 mm.

Shown in Figure 15 is the setup for this protocol. The median overall accuracy for this protocol is 3.01

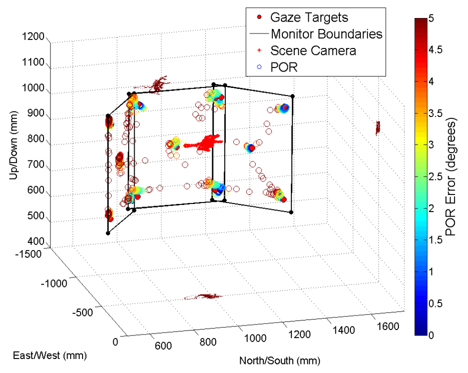


Figure 15: The experimental setup and data for Protocol 5.

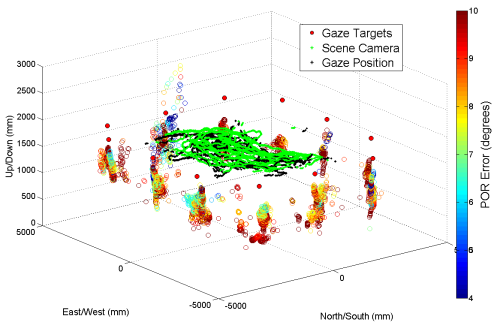


Figure 16: Wandering and gazing at 'hotspots' (POR error is arbitrary).

degrees which boils down to 103.80 pixels given the distance of the subject from the monitors.

4.2. Targets in 3D Space

In the following procedures we show how MobiGaze may be used to estimate PORs in 3D space, quite apart from a monitor. This is of particular advantage in MoBI experiments which tend to be more interactive, rather than passive experiences for subjects. In the case of tracking PORs on a computer monitor, MobiGaze will actually estimate where in the space of the monitor the subject is gazing. In the following protocol, MobiGaze will instead report as to whether or not a subjects POR points toward the vicinity a particular, pre-recorded points in 3D space. The tolerance (in degrees) for how near the gaze must be to the target is settable.

Protocol 4: Hotspots. In the experiment illustrated in 16, 12 'hotspots' are placed at roughly equal intervals on the walls of the lab. The subject wanders around the room gazing from hotspot to hotspot. In this configuration, MobiGaze determines whether or not a subject's gaze intersects with each hotspot by measuring the angle between their vectors. If the angle is less than a certain tolerance, we say that the subject is gazing toward the hotspot. We do not provide a measure of accuracy in this protocol as this would be somewhat artificial. But, this procedure for discrete determination of gaze intersection can be a very powerful tool in the context of a MoBI experiment. It allows researchers to determine with authority whether or not a subject is directing his gaze in the vicinity of any one of a number of different points in a room.[5]

5. Future Work

The software applications associated with MobiGaze were developed using the Embarcadero development environment which places them firmly in the realm of Windows only. It is desirable to repackage the software to make it cross platform, probably using Qt (`http://www.qt.io/`) as a GUI building environment. Such a project also requires incorporating the video grabbing software stack for each supported operating system.

The MobiGaze system is also designed to work in conjunction with the PhaseSpace motion capture system. In theory, however, any 3D motion capture technology could be used. This would most likely require some modification of the code in order to interface correctly with other systems' spatial orientation and units of measure. A desirable feature would be a GUI so that users rather than developers could adjust these parameters on the fly.

As of now, the MobiGaze cameras must be physically connected to the computer that runs GazeStream. The driver box, however, can be powered by a 5-V USB battery, but it is certainly desirable to develop a wireless capability for streaming the camera's ouput as well.

It is also true that sometimes in the course of an experiment MobiGaze glasses may change their position on the subjects face – either through slipping or by unconscious adjustment on the part of the wearer. This will, of course cause inaccuracies

[5]Incidentally, one may also configure hotspots so that the ray of interest is the one formed by two points in mocap space (not necessarily in MobiGaze space). This means that we may also reliably detect if a subject is point toward any point.

in the final output data. We may therefor want to improve the 'fit' by using the corneal reflection as a reference point, a method suggested in <9>.

6. Conclusions

We present MobiGaze, a modular, extensible, and open source eye tracking system developed at the Swartz Center for Computational Neuroscience for use in MoBI experiments. The pupil tracking and POR mapping procedures (which make use of geometric models) are detailed. We also present preliminary results indicating that the accuracy of the system is comparable to previous eye tracking systems. It is shown that the MobiGaze system is very flexible and is notable for its ability to accurately track a subject's gaze in ambulatory scenarios.

References

[1] S. Makeig, K. Gramann, T.-P. Jung, T. J. Sejnowski, H. Poizner, Linking brain, mind and behavior, International Journal of Psychophysiology 73 (2) (2009) 95–100.

[2] R. Newman, Y. Matsumoto, S. Rougeaux, A. Zelinsky, Real-time stereo tracking for head pose and gaze estimation, in: Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on, IEEE, 2000, pp. 122–128.

[3] A. Villanueva, R. Cabeza, S. Porta, Eye tracking: Pupil orientation geometrical modeling, Image and Vision Computing 24 (7) (2006) 663–679.

[4] E. D. Guestrin, M. Eizenman, General theory of remote gaze estimation using the pupil center and corneal reflections, Biomedical Engineering, IEEE Transactions on 53 (6) (2006) 1124–1133.

[5] Z. Zhu, Q. Ji, K. P. Bennett, Nonlinear eye gaze mapping function estimation via support vector regression, in: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, Vol. 1, IEEE, 2006, pp. 1132–1135.

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

[7] J. S. Babcock, J. B. Pelz, Building a lightweight eyetracking headgear, in: Proceedings of the 2004 symposium on Eye tracking research & applications, ACM, 2004, pp. 109–114.

[8] D. Li, J. Babcock, D. J. Parkhurst, openeyes: a low-cost head-mounted eye-tracking solution, in: Proceedings of the 2006 symposium on Eye tracking research & applications, ACM, 2006, pp. 95–100.

[9] F. J. Parada, D. Wyatte, C. Yu, R. Akavipat, B. Emerick, T. Busey, Experteyes: Open-source, high-definition eyetracking, Behavior research methods 47 (1) (2014) 73–84.

[10] A. Fitzgibbon, M. Pilu, R. B. Fisher, Direct least square fitting of ellipses, Pattern Analysis and Machine Intelligence, IEEE Transactions on 21 (5) (1999) 476–480.

[11] R. Bullock, Least-squares circle fit, URL:www.dtcenter.org/met/users/docs/write_ups/circle_fit.pdf.

[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] J. A. Nelder, R. Mead, The downhill simplex algorithm, Computer Journal 7 (S 308).

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical recipes in C, Vol. 2, Cambridge university press Cambridge, 1996.

[15] C. B. Duane, Close-range camera calibration, Photogram. Eng. Remote Sens 37 (1971) 855–866.

[16] J. H. Challis, A procedure for determining rigid body transformation parameters, Journal of biomechanics 28 (6) (1995) 733–737.

[17] K. Holmqvist, M. Nyström, F. Mulvey, Eye tracker data quality: what it is and how to measure it, in: Proceedings of the symposium on eye tracking research and applications, ACM, 2012, pp. 45–52.

[18] S. Nevalainen, J. Sajaniemi, Comparison of three eye tracking devices in psychology of programming research, 6th Annual Psychology of Programming Interest Group (2004) 170–184.

[19] D. W. Hansen, Q. Ji, In the eye of the beholder: A survey of models for eyes and gaze, Pattern Analysis and Machine Intelligence, IEEE Transactions on 32 (3) (2010) 478–500.

[20] A. Duchowski, Eye tracking methodology: Theory and practice, Vol. 373, Springer Science & Business Media, 2007.

[21] B. Noris, J.-B. Keller, A. Billard, A wearable gaze tracking system for children in unconstrained environments, Computer Vision and Image Understanding 115 (4) (2011) 476–486.

[22] J. W. Lee, C. W. Cho, K. Y. Shin, E. C. Lee, K. R. Park, 3d gaze tracking method using purkinje images on eye optical model and pupil, Optics and Lasers in Engineering 50 (5) (2012) 736–751.