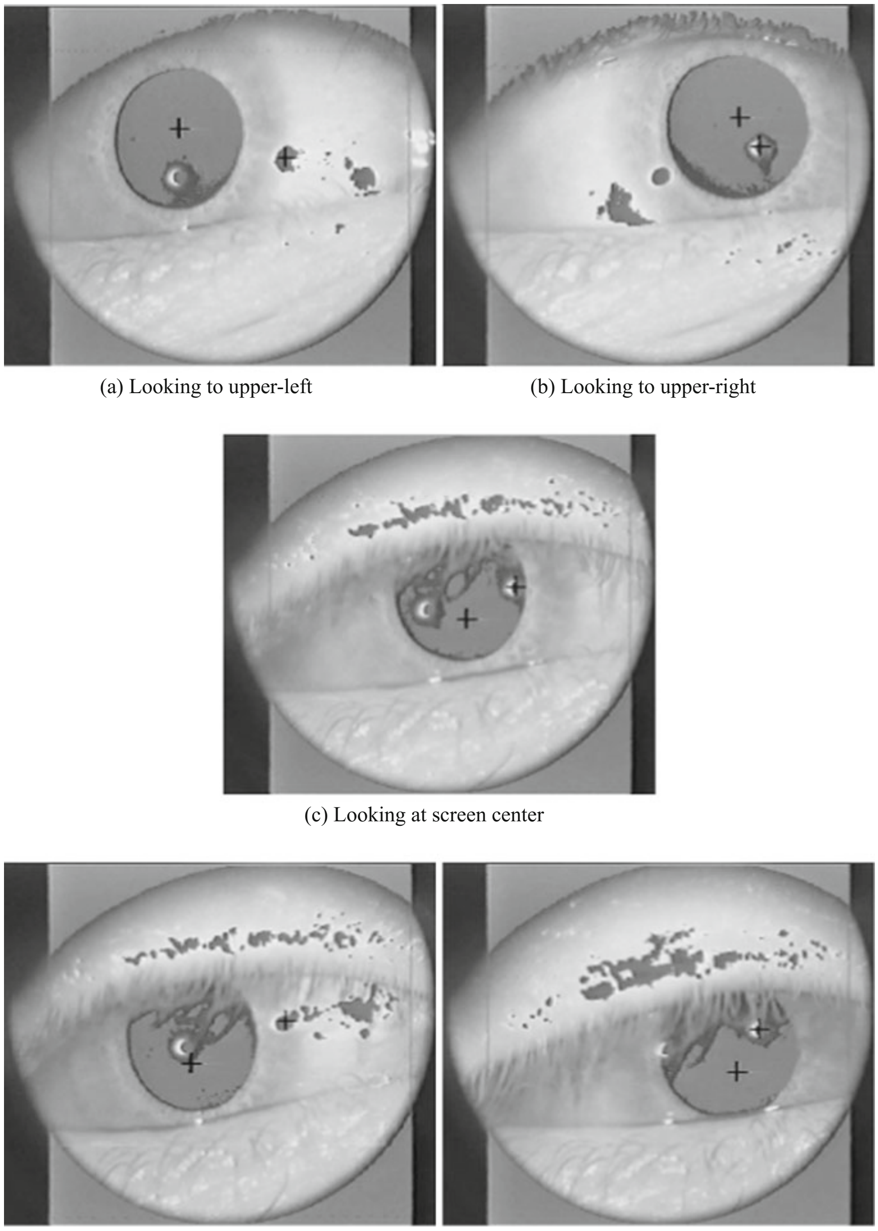
Chapter 8
Head-Mounted System Calibration

Currently, most video-based eye trackers require calibration. This is usually a sequence of simple stimuli displayed sequentially at far extents of the viewing region. The eye tracker calculates the point of regard (POR) by measuring the relative observed position of the pupil and corneal reflection at these locations, and then (most likely) interpolates the POR value at intermediate eye positions. The individual stimuli used for this purpose are simple white dots or crosshairs on a black background. In cases where the eye tracker is used in an outside setting (e.g., for use during driving or while walking outside the lab), calibration marks may be made from simple targets such as tape or other visible markers fixed to objects in the environment. The purpose of calibration is to present a sequence of visible points at fairly extreme viewing angle ranges (e.g., upper-left, upper-right, lower-left, lower-right). These extrema points should be chosen to provide a sufficiently large enough coordinate range to allow the eye tracker to interpolate the viewer's POR between extrema points. Most (video-based) eye trackers provide built-in calibration techniques where a number of such extrema points (e.g., three, five, or nine typically) are presented in order.

Apart from selection and/or presentation of properly distributed calibration stimulus points, a secondary but equally important goal of calibrating the eye tracker is correct adjustment of the eye tracker's optics and threshold levels to allow the device to properly recognize critical visual elements of the eyes. In the case of a video-based corneal reflection eye tracker, the device attempts to automatically detect the eye's pupil center and the center of the corneal reflection(s) (see Sect. 5.4). Corresponding to these elements, the eye tracker software may offer a mechanism to adjust both pupil and corneal reflection detection thresholds. Both must be set so that the eye tracker can easily detect the centers of the pupil and the corneal reflection, without either losing detection of either or confusing these targets with distracting artifacts such as eyelashes, rims of glasses, or contact lenses, to name a few usual problematic features. Figure 8.1 shows the eye images as seen by the eye tracker. Notice that the eye tracker has correctly labeled (with black crosshairs) the pupil center and one


(a) Looking to upper-left

(b) Looking to upper-right

(c) Looking at screen center

(d) Looking to bottom-left

(e) Looking to bottom-right

Fig. 8.1 Eye images during calibration (binocular eye tracking HMD)

of two[1] corneal reflections while the subject was viewing each of five calibration targets. Figure 8.1a shows a fairly good eye image with properly identified pupil and corneal reflections; notice also the pupil threshold "bleeding" that is seen on the sclera to the bottom and left of the pupil. The pupil threshold may have been set a touch lower to eliminate this artifact.

Figure 8.1d, e show potential problems caused by long eyelashes. Eyelashes (particularly with heavy application of mascara), contact lenses, and eyeglasses may pose problems by providing reflecting surfaces in the scene that may interfere with the eye tracker's ability to locate the corneal reflection. However, eye tracking technology is improving; better image processing techniques, such as facial recognition, are helping to overcome these common calibration problems.

The general calibration procedure is composed of the following steps, to be performed by the operator after starting the system display/tracking program.

1. Move the application window to align it with the eye tracker's central calibration dot.
2. Adjust the eye tracker's pupil and corneal reflection threshold controls.
3. Calibrate the eye tracker.
4. Reset the eye tracker and run (program records the data).
5. Save recorded data.
6. Optionally calibrate again.

The final calibration step may be repeated to judge the amount of instrument slippage during the experimental trial (see Sect. 8.2 below). If the display/tracking program is one that relies on serial communication with the eye tracker (i.e., the program is responsible for generating the calibration stimulus points) the program must display the calibration stimulus at the same location the eye tracker is currently presenting to the viewer. This rather crucial application program requirement is discussed in the following section.

8.1 Software Implementation

If the application program is responsible for displaying the visual stimulus (i.e., the calibration dots and then later the test imagery) it is imperative that: (a) the program knows when to display the calibration dots or the test imagery, and (b) the calibration dots are aligned as precisely as possible so that the program's calibration dot is displayed at the location where the eye tracker expects it to be. The latter condition is satisfied partly by appropriate mapping of coordinates between application program window (viewport) coordinates and the eye tracker screen coordinates, and the initial

[1]The eye tracker used to generate the images of Fig. 8.1 is a binocular eye tracker mounted inside a Head-Mounted Display (HMD). Due to the proximity of the optics to each eye, two infra-red Light Emitting Diodes (LEDs) are used on either side of the eye. This is somewhat unusual; typically (for table-mounted, or remote eye trackers) one LED or other infra-red light source is used to project a single corneal reflection.

placement of the application program window on the display screen (e.g., the window is moved to the appropriate position on the display device such as the TV or within the HMD). The former condition is met by obtaining a status word from the eye tracker itself. It is therefore imperative that the eye tracker provide this valuable piece of data along with the POR value(s).

Assuming the eye tracker provides its current state (along with the current eye coordinates x and y), the pseudocode for the usual graphics drawing routine, modified to draw the calibration points at the appropriate time, is shown in Listing 8.1. This routine is sensitive to the three possible modes of the eye tracker: RUN, RESET, and CALIBRATE. A typical double-buffered display update routine, for systems without eye trackers, would normally just be expected to set up the scene projection and object transformation matrices, and then display the scene (stimulus) and swap buffers. In the modified routine, what is mostly needed is a way of drawing the calibration stimulus at the appropriate time.

```
sceneProjection();              // e.g., ortho or perspective
sceneMatrix();           // model (object) transformation(s)
switch( eye tracker state ) {
  case RUN:
    if(displayStimulus)
      displayStimulus();            // show image, VR, etc.
    break;
  case RESET:
  case CALIBRATE:
    drawCalibrationDot($x-5, y-5, x+5, y+5$);
    break;
}
swapbuffers();                                // swap buffers
```

Listing 8.1 Graphics draw/expose routine augmented with mode-sensitive eye tracking code

Notice that in the pseudocode of Listing 8.1 the calibration stimulus is displayed in both RESET and CALIBRATE states. This facilitates the initial alignment of the application window with the eye tracker's initial positioning of its crosshairs. The default position is usually the center of the eye tracker screen. Provided the coordinate mapping routine is in place in the main loop of the program, the application program should correspondingly display its calibration dot at the center of its application window. The operator then simply positions the application window so that both points align in the eye tracker's scene monitor.

Notice also in Listing 8.1 that the stimulus scene (the image or VR environment) is only displayed if a display stimulus condition is satisfied. This condition may be set outside the drawing routine (e.g., in the main program loop) depending on some timing criterion. For example, if the experimental trial requires that an image be displayed for only five seconds, a timer in the main loop can be used to control the duration of the displayed stimulus by setting the display condition just after calibration has completed (state change from calibration to run). The timer then unsets the condition after the required duration has expired.

For VR applications, the draw routine may be preceded by viewport calls that determine which display, left or right, is drawn to. If the view is shifted horizontally, a binocular display is presented, otherwise, a biocular display is seen by the user. In VR, the calibration stimulus may require an orthographic projection call so that the 2D calibration points are not perturbed by a perspective transformation.

The main loop of the program is responsible for:

- Reading the data from the eye tracker (and the head tracker if one is being used)
- Mapping the eye tracker coordinates (and head tracker coordinates if one is being used)
- Starting/stopping timers if the experimental conditions call for a specific stimulus duration period
- Either storing or acting on the 2D (or 3D) gaze coordinates, if the application is diagnostic or gaze-contingent in nature, respectively

This general algorithm is shown as pseudocode in Listing 8.2. In this instance of the main loop, a timer of length DURATION is used to control the duration of stimulus display.

```
while(true) {
  getEyeTrackerData(x,y);                    // read serial port
  mapEyeTrackerData(x,y);                    // map coordinates
  switch( eye tracker state ) {
    case RUN:
      if(!starting) {
        starting = true;
        startTimer();
        displayStimulus = true;
        redraw();                            // redraw scene event
      }
      if(checkTimer() > DURATION) {
        displayStimulus = false;
        redraw();                            // redraw scene event
      } else {
        storeData(x,y);
      }
      break
    case RESET:
    case CALIBRATE:
      starting = false;
      redraw();                              // redraw scene event
      break;
  }
}
```

Listing 8.2 Main loop (2D imaging application)

A similar loop is needed for a VR application. There are a few additional requirements dealing with the head position/orientation tracker and calculation of the three-dimensional gaze vector. The pseudocode for the main loop of a VR application is shown in Listing 8.3. The main loop routine for the VR application mainly differs in the calculation of the gaze vector, which is dependent on the stereoscopic geometry of the binocular system. The only other main difference is the data storage requirements. Instead of just the 2D point of regard, now the locations of both eyes (or the gaze vector) and/or the head need to be recorded. For gaze-contingent applications, instead of data storage, the gaze vector may be used directly to manipulate the scene or the objects within.

8.2 Ancillary Calibration Procedures

The calibration procedure described above is crucial for proper operation of the eye tracking device. Some devices will not even generate an eye movement data word until they have been calibrated. Device calibration is therefore necessary for any subsequent eye tracker device operation. Additional calibration routines may be used to test the device accuracy slippage before and after experimental trials or perhaps for calibration of subsidiary software measures, possibly internal to the application program. Denoting the device calibration procedure as external, two ancillary procedures are presented, denoted internal, because these procedures are more related to the application software rather than to the eye tracker itself. The first such ancillary internal calibration procedure is described for checking the accuracy slippage of the eye tracker. This optional procedure is described for the 2D eye movement recording program described above. The second ancillary internal calibration procedure is described for 3D software calibration, as used in VR when binocular (vergence) eye movement parameters are not measurable a priori.

8.2.1 Internal 2D Calibration

Each experimental trial includes three calibration steps. Calibration is divided into two procedures: *external* and *internal*. External calibration pertains to the proprietary eye tracker instrument calibration procedure specified by the manufacturer. Internal calibration pertains to the procedure developed within the developed application system for measurement of eye tracker accuracy. The eye tracker is externally calibrated using the vendor's proprietary 5- or 9-point calibration procedure. Internal calibration can be developed over any number of points, e.g., 30. The layout of the internal calibration points, denoted by the symbol $+$ is shown in Fig. 8.2. The point arrays are framed for clarity; horizontal and vertical lines do not appear during calibration. Figure 8.2 also shows the position of external calibration points overlaid on top of internal calibration points (in this case 9 points depicted by circles). Internal cali-

```
while(true) {
  getHeadTrackerData(eye,dir,upv);           // read serial port
  getEyeTrackerData($x_l$,$y_l$,$x_r$,$y_r$);            // read serial port
  mapEyeTrackerData($x_l$,$y_l$,$x_r$,$y_r$);             // map coordinates

  // calculate linear gaze interpolant parameter s
  $s = b/(x_l - x_r + b)$;

  // set head position
  $\mathbf{h} = \left[eye_x, eye_y, eye_z\right]$;

  // calculate central view vector
  $\mathbf{v} = [(x_l + x_r)/2 - x_h, (y_l + y_r)/2 - y_h, f - z_h]$;

  // multiply $\mathbf{v}$ by FOB matrix
  transformVectorToHeadReferenceFrame($\mathbf{v}$);

  // calculate gaze point
  $\mathbf{g} = \mathbf{h} + s\mathbf{v}$;
  switch( eye tracker state ) {
    case RUN:
      if(!starting) {
        starting = true;
        startTimer();
        displayStimulus = true;
        redraw();                          // redraw scene event
      }
      if(checkTimer() > DURATION) {
        displayStimulus = false;
        redraw();                          // redraw scene event
      } else {
        storeData($x_l$,$y_l$,$x_r$,$y_r$);
      }
      break
    case RESET:
    case CALIBRATE:
      starting = false;
      redraw();                            // redraw scene event
      break;
  }
}
```

Listing 8.3 Main loop (3D virtual reality application)

bration is performed twice, immediately after external calibration (before stimulus display), and immediately after the stimulus display. Hence internal calibration is used to check the accuracy of the eye tracker before and after the stimulus display, to check for instrument slippage. The 30 internal calibration points can be shown in random order in a semi-interactive manner similar to the external calibration procedure. As each point is drawn on the screen (the subject is presented with a suitable simple

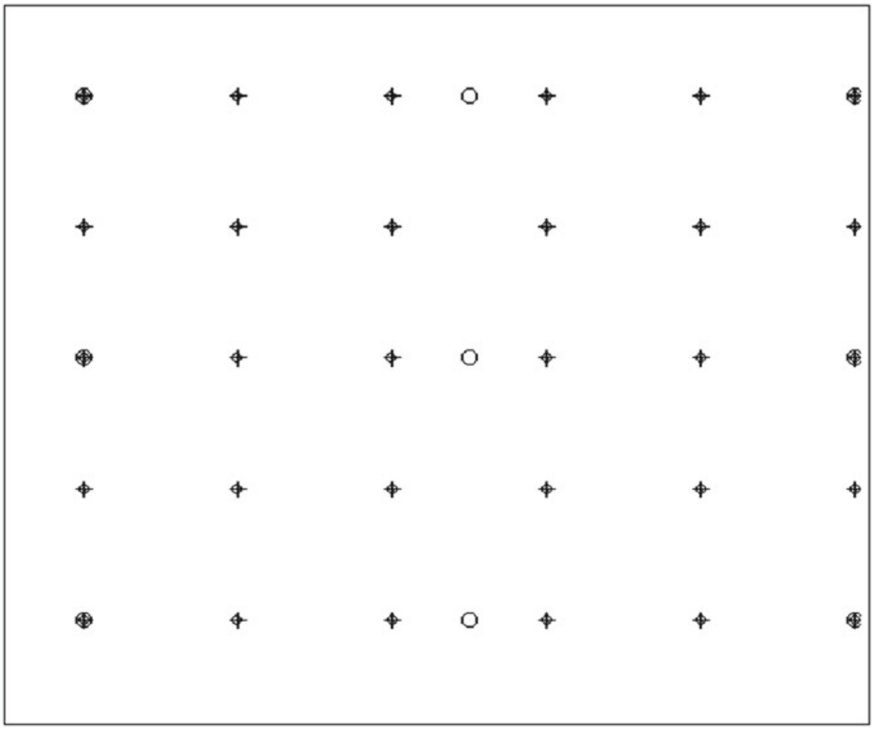


Fig. 8.2 Calibration stimulus: external (eye tracker) calibration points (*circles*) overlaid on internal calibration points (*crosses*)

stimulus, e.g., an $\times$ to minimize aliasing and flicker effects of an analog display), the application system waits for an input keypress before sampling eye movement data for a given sampling period (e.g., 800 ms). The input key delay allows the operator to observe eye stability on the eye tracker's eye monitor. The eye is judged to be stable once the eye tracker has repositioned the eye in the center of the camera frame. Recorded POR data is mapped to image coordinates in real-time. Calibration data can be stored in a flat text file for later evaluation.

Internal calibration procedures provide the basis for two statistical measures: the overall accuracy of the eye tracker, and the amount of instrument slippage during stimulus viewing. The latter measurement gives an indication of the instrument accuracy during the viewing task, i.e., by recording loss of accuracy between the before- and after-viewing calibration procedures. A simple experiment illustrates these measures. An average of recorded POR data points (centroid; following coordinate mapping) is obtained and the error between the centroid and calibration point is calculated. Each two-dimensional Euclidean distance measurement is converted to the full visual angle dependent on the viewing distance and calculated resolution of the television screen. A graphical example of this measurement is shown in Fig. 8.3.

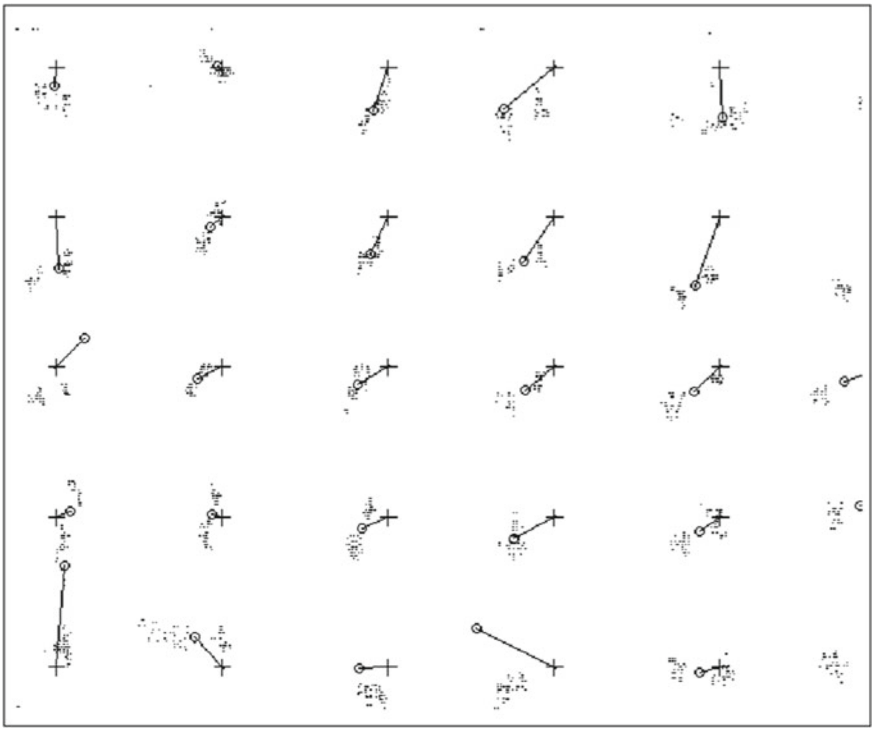


Fig. 8.3 Typical per-trial calibration data (one subject) after stimulus viewing (avg. error: 1.77°)

The internal calibration locations are represented by $+$, sample measurements are represented by individual pixel dots, and centroid gaze positions are represented by circles, joined with the corresponding calibration point by a line. The length of the line is the average error deviation in pixels.

To quantify overall eye tracker accuracy succinctly, the average calibration error can be obtained from each set of calibration points in order to calculate an overall average statistic for the eye tracker. The resulting average instrument error is an average statistic over all calibration runs.

To quantify instrument slippage, statistical analysis can be performed on before- and after-viewing mean error measure. Figure 8.4 shows a composite plot of before and after internal calibration stimulus display. Notice that, in this example, measured eye positions generally correspond well to calibration points. To quantify this correspondence, a one-way ANOVA can be performed on the means of the before- and after-viewing average error measures. If no significant difference is reported between the means, one can consider instrument slippage to be nominal.

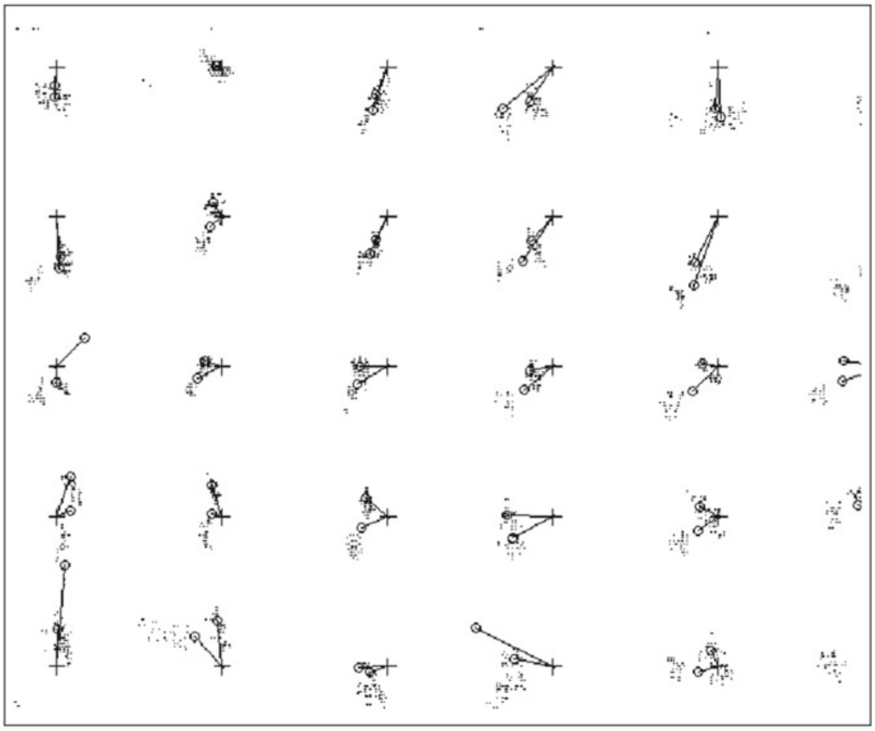
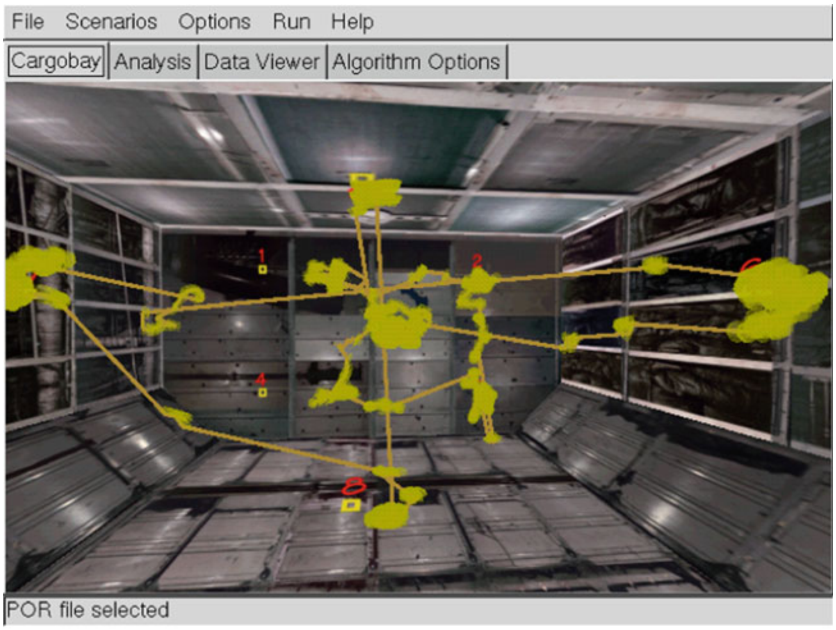


Fig. 8.4 Composite calibration data showing eye tracker slippage

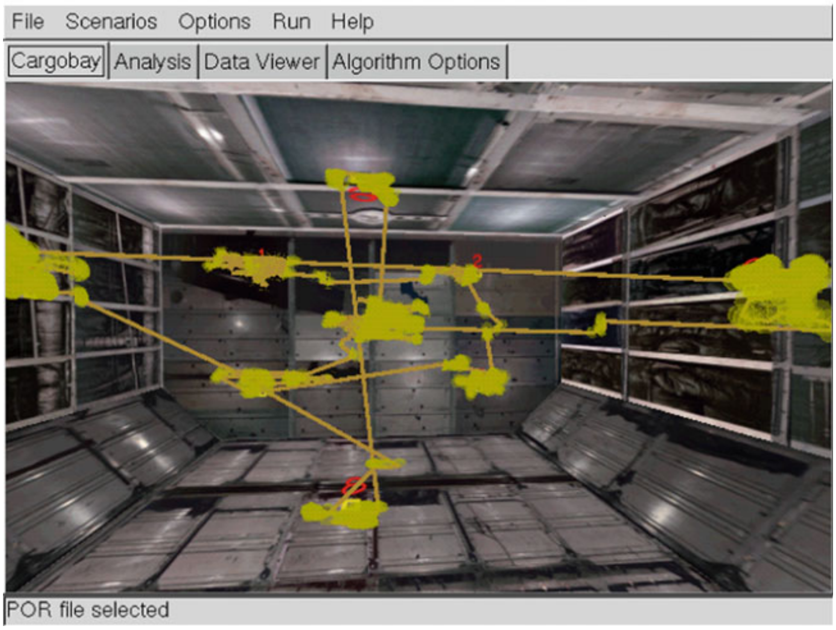
8.2.2 Internal 3D Calibration

For 3D gaze point estimation, determination of the scalar s (dependent on interpupillary distance, or baseline b) and focal distance f used in Eqs. (7.18)–(7.20) in Sect. 7.3 is difficult. Interpupillary distance cannot easily be measured in VR because the left and right eye tracking components function independently. That is, there is no common reference point. An internal 3D calibration procedure can be designed to estimate the interpupillary distance scaling factor s empirically (Duchowski et al. 2001).

The 3D calibration relies on a specially marked environment, containing nine clearly visible fixation targets, illustrated in Fig. 8.5. Figure 8.5 shows scanpaths in a 3D virtual reality environment prior to eye movement analysis (see Chap. 13). Green spheres in the figure represent raw GIP data. The nine $\times$ targets are distributed on five walls of the environment to allow head movement to be taken into account during analysis. Without a precise estimate of b and f, computed Gaze Intersection Points (GIPs) may appear stretched or compressed in the horizontal or vertical direction, as shown in Fig. 8.5a. Following manual manipulation of b and f values, GIP data are recalculated and displayed interactively. The goal is to align the calculated GIP



(a) Prior to adjustment



(b) Following adjustment

Fig. 8.5 Adjustment of left and right eye scale factors

locations with the environmental targets on which the user was instructed to fixate during calibration. An example of this type of adjustment is shown in Fig. 8.5b. Notice that the GIPs (represented by green transparent spheres) are now better aligned over the red targets than the raw data in Fig. 8.5a. Once determined, appropriate scale factors are used to adjust each participant's eye movement data in all subsequent trials.

8.3 Summary and Further Reading

In summary, calibration of the eye tracker is essential toward proper eye movement data collection and analysis. The calibration procedure typically involves proper setting of the eye tracker's imaging optics (e.g., eye camera focus and contrast) and software threshold levels (e.g., pupil and corneal reflection). Although today's eye trackers have improved imaging and image processing components, traditional sources of calibration errors still exist and continue to pose problems. For example, contact lenses or eyeglasses may still interfere with some eye tracking devices; long eyelashes, heavy eye makeup, or "droopy" eyelids may prevent proper imaging of the eye, and in some scenarios the subject's own body and head movement may also cause imaging problems. In some instances, head stabilization (e.g., the use of a chin rest) may be required.

Unfortunately, there is no widely accepted "how-to" text explaining proper calibration procedures. Most often, the eye tracker manufacturer's usage manual is the best primary source of information. Furthermore, by far the best advice for good manual calibration is experience and practice. Proper usage of the eye tracker, and in particular fast and "good" calibration, often comes with repeated use of the device. A current goal of eye tracking research is to devise a system for which calibration is not necessary; i.e., manufacturers are currently pursuing the development of an autocalibrating eye tracker. Although the current goal of doing away with calibration altogether has not yet been achieved, there is probably no better substitute for proper manual calibration than hands-on experience. It is often the case that new eye tracker operators are uncomfortable with the various controls of the device and may also feel nervous with directing human subjects to participate in eye tracking studies. However, with practice, one can quickly learn to properly use and calibrate an eye tracker. Once a sufficient comfort level is reached, proper calibration can be performed within a matter of seconds.