HAND-EYE COORDINATION IN DESKTOP VIRTUAL REALITY

THEODORE W. HALL
Chinese University of Hong Kong

Abstract. For hand-eye coordination and intuitive interaction with virtual-reality displays, the projected image of a 3-D cursor in virtual space should correspond to the real position of the 3-D input device that controls it. This paper summarizes some of the issues and algorithms for coordinating the physical and virtual worlds.

1. Introduction

Contrary to Hollywood fantasies, virtual reality is not an out-of-body experience. Humans experience inertia and gravity and have a proprioceptive sense of body posture that is independent of vision. Moreover, many virtual-reality displays augment the user's view of the real world but do not completely mask it out or replace it. Thus, intuitive control and realistic interaction with virtual reality depend on accurate hand-eye coordination: the projected image of a 3-D cursor in virtual space should align visually with the real position of the 3-D input device that controls it. Though this may seem obvious, it is not automatic. Position input and graphic output rely on separate devices with independent coordinate references. Some systems provide neither the hardware nor the software necessary to map coordinates from one reference to the other, leaving the application programmer or end user to fend for themselves.

In the remainder of this paper, "desktop display" is short-hand for any non-head-mounted display. The discussion applies as well to wall-mounted displays. "Stereo glasses" are eye wear for filtering a desktop display into separate left and right views. These must also be equipped for position tracking. "3-D mouse" refers to a hand-held pointing device in a three-dimensional position tracking system. This may be a simple point-and-click device similar to the common 2-D mouse, or it may be incorporated into a more sophisticated system such as a data glove. "Tracker" refers to the reference device in the position-tracking system. The system reports the positions of the stereo glasses and 3-D mouse relative to this device.

2. Calibrating the Tracking System

The first step toward achieving any visible correspondence between the physical pointer and the virtual cursor is to calibrate the position tracking system relative to the display. For desktop displays, this presumes that the tracker is mounted in a stable position relative to the display, and that the display is within the tracker's operating range. The precise position of the tracker is arbitrary, and generally involves rotation and translation along all three axes. Moreover, the display hardware itself may be maladjusted, resulting in slightly different scale factors (pixels per millimeter) on the x

R. Junge (ed.), CAAD Futures 1997, 177-182.

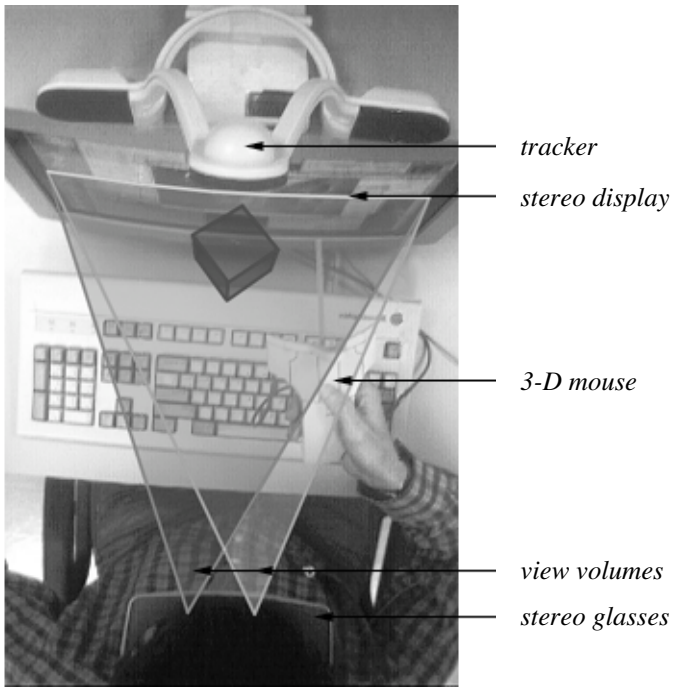

Figure 1: Correspondence of physical and virtual view volumes.

and *y* axes.

Figure 1 shows a plan view of a typical desktop virtual-reality system. The tracker is fastened to the top of the display (in this case, with adhesive tape). It monitors the positions of the 3-D mouse and stereo glasses relative to its own coordinate axes, and reports them to the application via the computer's serial ports. The application must derive and apply a transformation from tracker coordinates to display coordinates and thence to virtual world coordinates. These transformations are necessary to align the virtual 3-D cursor with the physical 3-D mouse. Similarly, the virtual projection plane and view points must correspond with the physical display plane and the user's eyes.

2.1. CALIBRATION PROCEDURE

In theory, three non-collinear points define a plane. In practice, most computer displays are not planar, but rather cylindrical or spherical. It's difficult to position three points on such a display without one of them bulging. Four points, located symmetrically at the center of each edge, avoid this problem. These points define horizontal and vertical vectors, parallel to the *x* and *y* axes. The cross product of these vectors defines the *z* axis and the orientation of the ideal plane. The location of the plane is such that the average *z* translation of the four points is zero.

For each of the four points, the calibration procedure draws a target and prompts the user to touch and click with the 3-D mouse. From the tracker coordinates reported for the targets, the procedure computes roll, pitch, and yaw rotations to bring the

tracker axes into alignment with the display axes.

After aligning the axes, the procedure computes display scale factors. The display image size is easily altered by anyone who adjusts the hardware controls, and may differ significantly from the nominal values reported in the documentation or obtained from standard system functions. The calibration procedure uses the position tracking system to measure the actual display dimensions (in millimeters), and uses those dimensions to compute accurate scale factors.

Finally, the procedure subtracts the rotated-and-scaled tracker coordinates from the display coordinates to determine translations.

The transformation from tracker coordinates to display coordinates (roll, pitch, yaw, scale, translate) is constant as long as the tracker remains in a stable position relative to the display and the display image size doesn't change. The application may save the transformation in a file and retrieve it in subsequent runs.

2.2. TRANSFORMATION FROM TRACKER TO DISPLAY

In the following, the subscript *mt* denotes coordinates of the mouse relative to the tracker, *td* denotes coordinates of the tracker relative to the display, and *md* denotes coordinates of the mouse relative to the display. Matrices are defined to post-multiply row vertices. (To pre-multiply column vertices, transpose the matrices and reverse the order of multiplication.)

Let $\mathbf{T}_{td}$ represent the transformation matrix formed from the roll, pitch, yaw, scale, and translation of the tracker relative to the display (determined above). This matrix transforms the translation of the 3-D mouse from tracker to display coordinates:

$$\begin{bmatrix} xt_{md} & yt_{md} & zt_{md} & 1 \end{bmatrix} = \begin{bmatrix} xt_{mt} & yt_{mt} & zt_{mt} & 1 \end{bmatrix} \times \mathbf{T}_{td}$$

It's often useful to "transform" the rotation as well – to factor it into roll, pitch, and yaw components on the display's *z*, *x*, and *y* axes. These derive from the mouse's rotated basis vectors. In its own coordinate system, its basis vectors are the rows of an identity matrix. In display coordinates, they are the rows of a rotation matrix.

Let $\mathbf{R}_{mt}$ represent the rotation of the mouse relative to the tracker. Let $\mathbf{R}_{td}$ represent the rotation of the tracker relative to the display. Then:

$$\begin{bmatrix} \mathbf{i}_{md} \\ \mathbf{j}_{md} \\ \mathbf{k}_{md} \end{bmatrix} = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \\ k_x & k_y & k_z \end{bmatrix} = \mathbf{R}_{md} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \mathbf{R}_{mt} \times \mathbf{R}_{td}$$

In general, assuming the order of rotation is *roll* about *z*, *pitch* about *x*, yaw about *y*:

$$\begin{aligned} roll_{md} &= \tan^{-1}\left(i_y / j_y\right) \\ pitch_{md} &= \tan^{-1}\left(-k_y \Big/ \sqrt{k_x^2 + k_z^2}\right) \\ yaw_{md} &= \tan^{-1}\left(k_x / k_z\right) \end{aligned}$$

In the special case that i_y and j_y are both zero, or (equivalently) k_x and k_z are both zero, then the pitch is $\pm\pi/2$ (opposite to the sign of k_y). The roll can be set to zero, and the yaw can be computed from i_z and i_x:

$$roll_{md} = 0$$

$$pitch_{md} = \begin{cases} \pi/2 & \text{if } k_y = -1 \\ -\pi/2 & \text{if } k_y = 1 \end{cases}$$

$$yaw_{md} = \tan^{-1}\left(-i_z / i_x\right)$$

3. Stereoscopic Perspective Projection

In essence, stereoscopic perspective is simply a double application of the well-known procedure for monoscopic perspective. Extend rays from a viewing point through points in a scene. The intersections of these rays with a projection plane define the perspective projection of the scene on that plane. The ray from the viewing point perpendicular to the plane defines the center of projection. Computed perspective derives algebraically from the proportions of similar triangles: it scales width and height from the center of projection, according to the ratio of distances from the

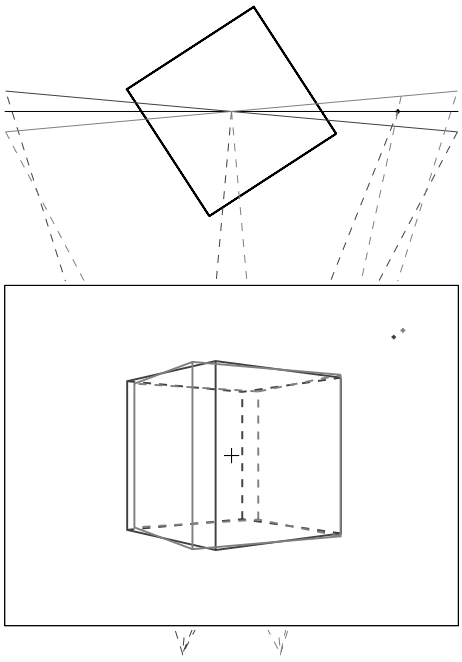

Figure 2: Incorrect construction of stereoscopic perspective (by rotation).

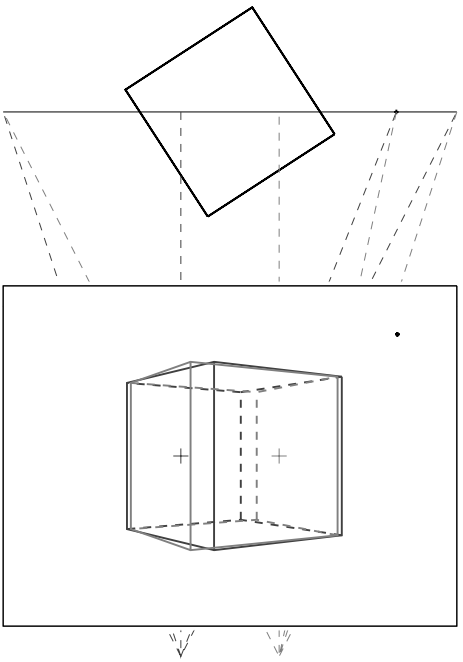

Figure 3: Correct construction of stereoscopic perspective (by translation).

viewing point to the projection plane and the point in the scene.

The center of vision can pan across the scene, from one ray to another, without any effect on the intersections of the rays with the plane – provided that the viewing point, the object, and the projection plane remain fixed in place.

Because the eyes rotate to converge their centers of vision on a point of interest, there is a temptation to compute stereoscopic perspective on the basis of a rotation between two views, as shown in Figure 2. This assumes that the eyes' centers of vision are also their centers of projection, resulting in two distinct, incongruous projection planes. It leads to vertical parallax between the two projections that should never occur as long as the eyes are level and equidistant from the display. Points in the plane of the display, which should project to the same pixel in both views, project to two different pixels. The error is small near the center, but becomes increasingly severe near the edges. These discrepancies confound depth perception and hinder hand-eye coordination in the virtual world.

Figure 3 shows the correct algorithm. Both views project onto a common plane, with only a horizontal offset between them. If one assumes a symmetric view frustum, then one must enlarge the horizontal angle and clip one side. Alternatively, one can

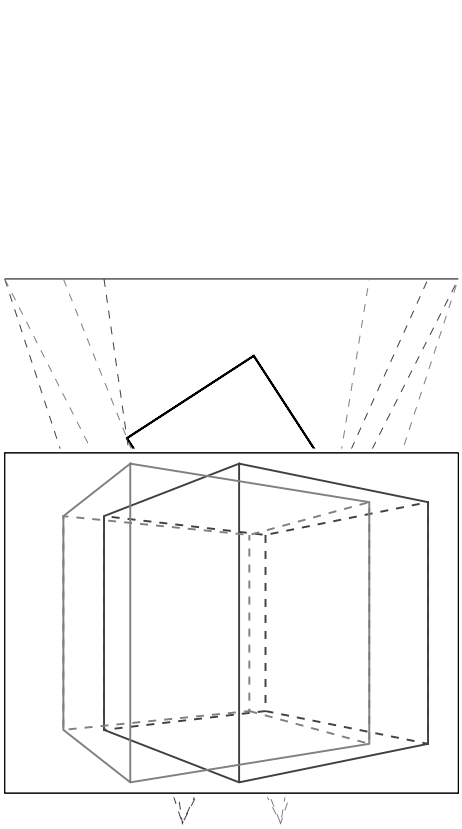

Figure 4: Moving the object closer.

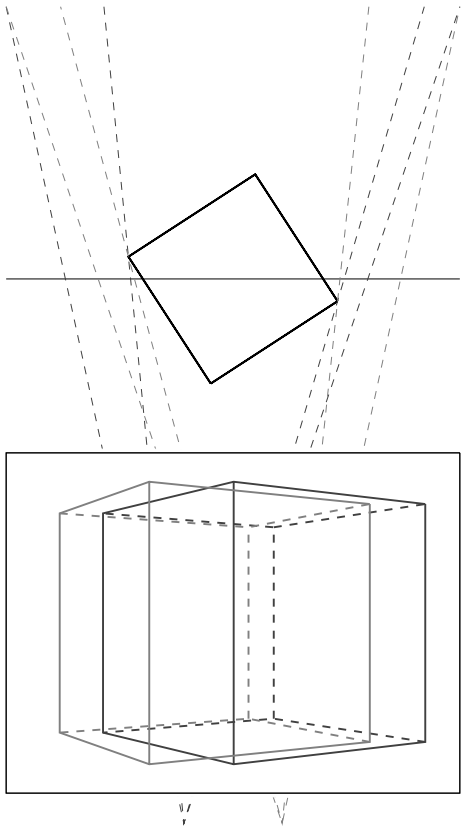

Figure 5: Zooming-in on the object.

modify the equations for perspective projection to account for the asymmetric frustum.

It's desirable to control the apparent depth of the stereo image, to avoid large discrepancies between optical focus and binocular convergence, as well as to insure that interactive elements in the virtual world appear within reach of the 3-D mouse.

Problems may arise in trying to interact with a small image of a large object: the interactive elements may be too small to resolve, let alone control. There are several strategies for enlarging the image.

Figure 4 shows the effect of moving the object closer to the view points. Though it enlarges the image, it also increases the parallax. The result still appears as a small three-dimensional object, but now uncomfortably close to the user's face.

Figure 5 shows the effect of zooming-in on the object, by reducing the view angle and scaling the image to fill the display. This scales the width and height of the stereo image, but not its depth. Depth cues from both perspective and parallax make the object appear flattened.

Figure 6 shows the effect of enlarging the object. Alternatively, if the distance between the view points is reduced in proportion to their distance from the projection plane, the effect appears *as if* the object has been enlarged. Thus, the visual scale of the virtual world derives from the placement of the view points and projection plane. These should conform to the view angles and proportions of the real view volume, defined by the user's eyes and display plane. The application may apply any scale to the view volume, but should apply it uniformly.

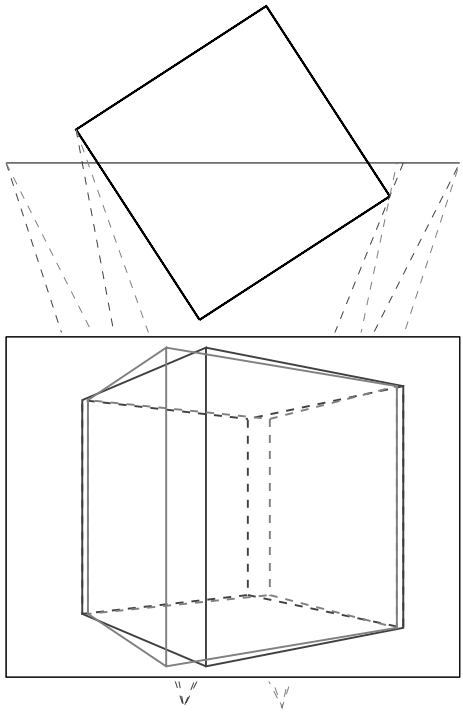

Figure 6: Scaling the object.