Virtual Trackballs Revisited

Knud Henriksen*, Jon Sporring†, and Kasper Hornbæk‡

June 30, 2002

Abstract

Rotation of 3D objects by a 2D mouse is a typical task in applications such as computer aided design, operation simulations, and desktop virtual reality. The most commonly used rotation technique is a virtual trackball surrounding the object and operated by the mouse pointer. The first but still popular virtual trackball was described by Chen *et al.* [1]. We show that the method by Chen *et al.* does not rotate the object along the intended great arc on the virtual trackball, and we give a correction. Another popular virtual trackball is Shoemake's quaternion implementation [2], which is shown to be a special case of Chen *et al.* Shoemake extends the scope of the virtual trackball to the full screen. Unfortunately, Shoemake's trackball is inhomogeneous and discontinuous with consequences for usability. Finally, we will discuss alternative extensions to virtual trackballs and their consequences for usability issues.

1 Virtual Trackballs

A virtual trackball is a device for controlling 3D rotations by moving a 2D mouse. Conceptually, a virtual trackball can be thought of as a 3D sphere with radius $r = |\boldsymbol{r}|$ located on the negative $\boldsymbol{z}-$axis behind the screen/image plane, see Figure 1.

Consider a virtual 3D sphere behind the image plane, and consider the tangent plane of that 3D sphere parallel to the image plane: The 2D motion of the mouse in the image plane has a one-to-one mapping to the tangent plane of the 3D sphere, which in turn may be mapped onto the visible half of the 3D sphere. In the following, the 2D image plane Π is imbedded in 3D space with its own coordinate system $\boldsymbol{xyz}$, so a 2D point $\boldsymbol{p}_a = (x_x, y_a)^\top$ in the image plane is thought of as a 3D point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$ in the imbedded image plane. The mapping from (the imbedded) image plane (screen) coordinates to the 3D sphere is specified by a function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$, typically orthographic project. By the motion of the mouse on the image plane, one may now determine the vitual motion of the 3D points on the sphere and let these determine the actual 3D rotation.

The idea is the following. Let the mouse be pressed at point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$ and moved to point $\boldsymbol{p}_c = (x_c, y_c, 0)^\top$ where it is released. That is, the mouse moved from point $\boldsymbol{p}_a$ along the displacement vector $\boldsymbol{d} = \boldsymbol{p}_c = \boldsymbol{p}_c - \boldsymbol{p}_a$. The corresponding points on the sphere are $\boldsymbol{P}_a = \boldsymbol{m}(\boldsymbol{p}_a)$ and $\boldsymbol{P}_c = \boldsymbol{m}(\boldsymbol{p}_c)$, and the sphere should be rotated about its center $\boldsymbol{O}$ along the great circular arc defined by the three points $\boldsymbol{O}, \boldsymbol{P}_a$, and $\boldsymbol{P}_c$.

In the following we discuss the three most popular virtual trackballs: Chen *et al.* [1], Shoemake [2], and Holroyd [20].

2 The Chen *et al.* Trackball

This section gives a review of the virtual trackball by Chen *et al.* [1]. Let the points $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ be arbitrary points in the image plane, and assume that the location vector $\boldsymbol{p}_a$ makes an angle φ with the $\boldsymbol{x}-$axis, and

*Henriksen is with the Department of Computer Science (DIKU), University of Copenhagen, Universitetsparken 1, DK–2100 Copenhagen Ø, Denmark

†Sporring is with 3DLab, School of Dentistry, University of Copenhagen, Nørre Alle 20, DK–2200 Copenhagen N, Denmark

‡Hornbæk is with the Natural Sciences ICT Competence Center, Universitetsparken 5, DK–2100 Copenhagen Ø, Denmark.

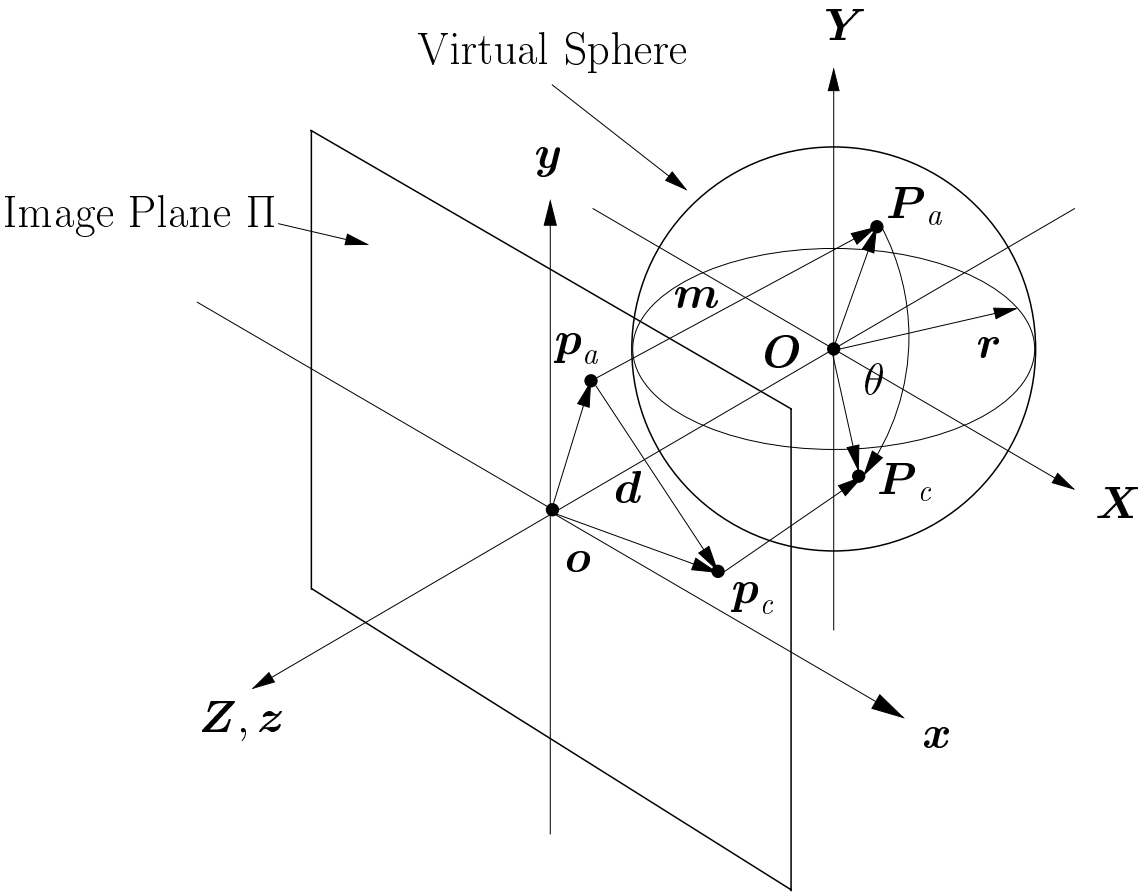


Figure 1: A virtual trackball can be thought of as a 3D sphere located behind the image plane (computer-screen). The points $\boldsymbol{p}$ on the image plane are mapped to points $\boldsymbol{P}$ on the 3D sphere by a function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$, i.e. $\boldsymbol{P} = \boldsymbol{m}(\boldsymbol{p})$.

that the displacement vector $\boldsymbol{d}$ makes an angle $\tau + \varphi$ with the $\boldsymbol{x}-$axis, i.e. the displacement vector $\boldsymbol{d}$ makes an angle τ with the location vector $\boldsymbol{p}_a$, see Figure 2.

The problem is to find a 3D rotation axis $\boldsymbol{u}$ through the center of the sphere $\boldsymbol{O}$ which rotates the point $\boldsymbol{P}_a = \boldsymbol{m}(\boldsymbol{p}_a)$ to the point $\boldsymbol{P}_c = \boldsymbol{m}(\boldsymbol{p}_c)$ along a great circular arc on the 3D sphere. Chen *et al.* find the axis of rotation $\boldsymbol{u}$ by first considering two special cases and then deriving the general case.

2.1 Deriving the Transformation

Case 1: The Point $\boldsymbol{p}_a$ is at the Origin $\boldsymbol{o}$

Consider the special case where the point $\boldsymbol{p}_a$ is located at the origin $\boldsymbol{o} = \boldsymbol{m}^{-1}(\boldsymbol{O})$, i.e. $\boldsymbol{p}_a = (0, 0, 0)^\top$, see Figure 3(a). The displacement vector $\boldsymbol{d}$ is the projection by $\boldsymbol{m}^{-1}$ of the great circular arc defined by the points $\boldsymbol{P}_a, \boldsymbol{P}_c$, and $\boldsymbol{O}$. In this special case the displacement vector $\boldsymbol{d}$ in the image plane has the coordinates

$$\boldsymbol{d} = (d_x, d_y, d_z)^\top = |\boldsymbol{d}|(\cos\tau, \sin\tau, 0)^\top \tag{1}$$

where $|\boldsymbol{d}|$ is the Euclidean length of the displacement vector. The rotation axis $\widetilde{\boldsymbol{u}} = (u_x, u_y, u_z)^\top$ is parallel to the image plane and perpendicular to $\boldsymbol{d}$. That is, the unit rotation axis is equal to

$$\widetilde{\boldsymbol{u}} = (-\sin\tau, \cos\tau, 0)^\top \tag{2}$$

Case 2: The Point $\boldsymbol{p}_a$ is on the $\boldsymbol{x}-$axis

Consider the special case where the point $\boldsymbol{p}_a$ is located on the $\boldsymbol{x}-$axis i.e. $\boldsymbol{p}_a = (x_a, 0, 0)^\top$, see Fig-

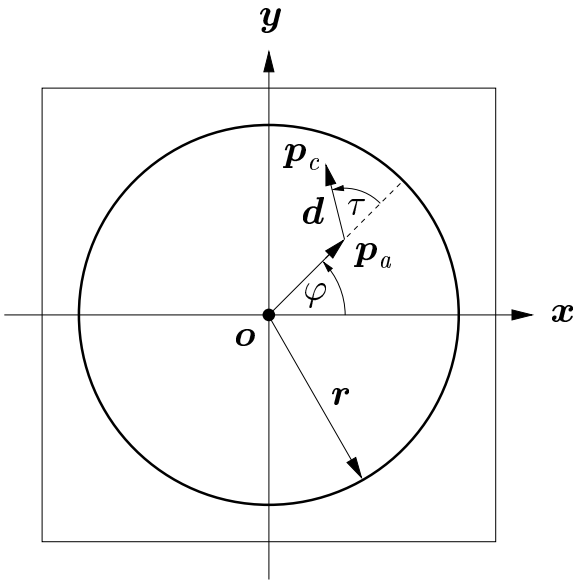


Figure 2: Screen view where the mouse is pressed at point $\boldsymbol{p}_a$ and moved to point $\boldsymbol{p}_c$ where it is released (the general case, case 3). The location vector $\boldsymbol{p}_a$ makes an angle φ with the $\boldsymbol{x}-$axis, and the displacement vector $\boldsymbol{d}$ makes an angle τ with the location vector $\boldsymbol{p}_a$.

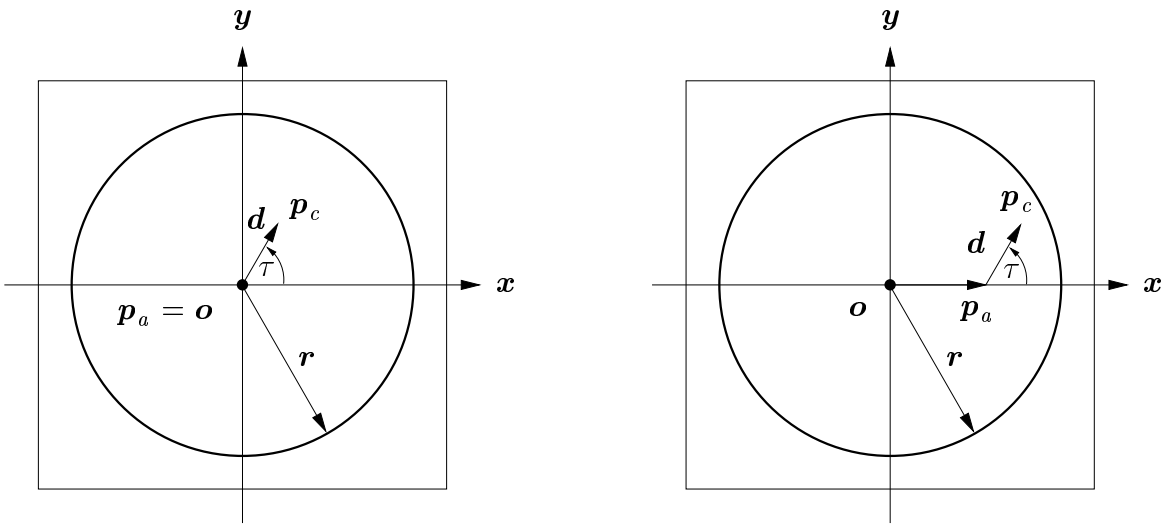


Figure 3: **(a)** Screen view of the special case where the point $\boldsymbol{p}_a$ is located at the origin, i.e. $\boldsymbol{p}_a = \boldsymbol{o} = (0,0,0)^\top$ (case 1). In this special case the location vector $\boldsymbol{p}_a$ makes an angle $\varphi = 0$ with the $\boldsymbol{x}-$axis, and the displacement vector $\boldsymbol{d}$ makes an angle τ with the location vector $\boldsymbol{p}_a$. **(b)** Screen view of the special case where the point $\boldsymbol{p}_a$ is located on the $\boldsymbol{x}-$axis(case 2). In this special case the location vector $\boldsymbol{p}_a$ makes an angle $\varphi = 0$ with the $\boldsymbol{x}-$axis, and the displacement vector $\boldsymbol{d}$ makes an angle τ with the location vector $\boldsymbol{p}_a$.

ure 3(b). In this case the rotation axis $\boldsymbol{u}$ from (2) is transformed such that the origin $\boldsymbol{o}$ becomes the point $\boldsymbol{p}_a$. This transformation consists of a rotation around the $\boldsymbol{y}-$axis by some angle ω, which will be specified later. That is, in this special case the unit axis of rotation $\widehat{\boldsymbol{u}}$ will be

$$\widehat{\boldsymbol{u}} = \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & \cos\omega \end{pmatrix} \begin{pmatrix} -\sin\tau \\ \cos\tau \\ 0 \end{pmatrix} \tag{3}$$

Unfortunately, this is incorrect, and will be discussed in detail in Section 2.2 and 2.3. For the sake of the review, the following continues the review of Chen *et al.*'s line of thought.

Case 3: The Point $\boldsymbol{p}_a$ is at a General Position

In the general case, where the point is neither at the origin nor on the $\boldsymbol{x}-$axis, it is rotated around the $\boldsymbol{z}-$axis by some angle φ, see Figure 2. The actual unit rotation axis can be obtained by rotating the axis, $\widehat{\boldsymbol{u}}$, the angle φ around the $\boldsymbol{z}-$axis. That is, the final unit rotation axis $\boldsymbol{u}$ becomes

$$\boldsymbol{u} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & \cos\omega \end{pmatrix} \begin{pmatrix} -\sin\tau \\ \cos\tau \\ 0 \end{pmatrix} \tag{4}$$

The angle φ is the angle of the location vector $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$ with the $\boldsymbol{x}-$axis. That is, the angle φ equals

$$\varphi = \tan^{-1}\left(\frac{y_a}{x_a}\right) \tag{5}$$

The angle ω might be computed in several ways. In the paper by [1] it is specified as a function $f : \mathbb{R} \longrightarrow \mathbb{R}$ of the distances $\mid \boldsymbol{p}_a \mid$ and $\mid \boldsymbol{r} \mid$ as follows, where $\boldsymbol{r}$ is the radius of the projection of the virtual sphere onto the image plane.

$$\omega = f\left(\frac{\mid \boldsymbol{p}_a \mid}{\mid \boldsymbol{r} \mid}\right) \tag{6}$$

where $f : \mathbb{R} \longrightarrow \mathbb{R}$ is a monotone function which satisfies

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \frac{\pi}{2} & \text{if} \quad x \geq 1 \end{cases} \tag{7}$$

By experiment Chen *et al.* have found that f is equal to

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \frac{\pi}{2}x & \text{if} \quad 0 \leq x \leq 1 \\ \frac{\pi}{2} & \text{if} \quad x \geq 1 \end{cases} \tag{8}$$

The actual rotation matrix, which rotates a point the angle θ around the axis $\boldsymbol{u} = (u_x, u_y, u_z)^\top$, (4), is given by [21, page 227]

$$\begin{pmatrix} u_x^2 + (1-u_x^2)\cos\theta & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_x u_y(1-\cos\theta) + u_z\sin\theta & u_y^2 + (1-u_y^2)\cos\theta & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_x u_z(1-\cos\theta) - u_y\sin\theta & u_y u_z(1-\cos\theta) + u_x\sin\theta & u_z^2 + (1-u_z^2)\cos\theta \end{pmatrix} \tag{9}$$

In the paper by [1] the angle θ is chosen by experiment to be

$$\theta = \frac{\pi}{2}\frac{\mid \boldsymbol{d} \mid}{\mid \boldsymbol{r} \mid}\left(1 - \left(1 - \frac{0.2}{\pi}\right)\frac{2\omega}{\pi}\left(1 - \mid \cos\tau \mid\right)\right) \tag{10}$$

where $\boldsymbol{d}$ is given by (1), ω is given by (6–8), and τ is as shown in Figure 2.

2.2 Remarks on Approach by Chen *et al.*

As mentioned, the rotation axis $\boldsymbol{u}$ computed by (4) is incorrect, since it is not necessarily perpendicular to the displacement vector $\boldsymbol{d}$, (1). In order to illustrate this problem, let the displacement vector $\boldsymbol{d}$ make a constant angle τ with the $\boldsymbol{x}-$axis, i.e. the angle between the location vector $\boldsymbol{p}_a$ and the displacement vector $\boldsymbol{d}$ makes the angle τ. That is, the displacement vector (1) becomes

$$\boldsymbol{d} = (d_x, d_y, d_z)^\top = d(\cos(\tau), \sin(\tau), 0)^\top \tag{11}$$

and the rotation axis, (4) becomes

$$\boldsymbol{u} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & \cos\omega \end{pmatrix} \begin{pmatrix} -\sin(\tau-\varphi) \\ \cos(\tau-\varphi) \\ 0 \end{pmatrix} \tag{12}$$

$$= \begin{pmatrix} -\cos\tau\,\sin\varphi - \cos\omega\,\cos\varphi\,\sin\tau \\ \cos\varphi\,\cos\tau - \cos\omega\,\sin\varphi\,\sin\tau \\ \sin\omega\,\sin(\tau-\varphi) \end{pmatrix} \tag{13}$$

To see that the displacement vector $\boldsymbol{d}$, (11) and the rotation axis $\boldsymbol{u}$, (13) are *not* perpendicular, the dot-product of the unit displacement vector $\boldsymbol{d}/|\boldsymbol{d}|$, (11) and the unit rotation axis $\boldsymbol{u}$, (13) is computed

$$\boldsymbol{d}\cdot\boldsymbol{u} = \boldsymbol{d}^\top\boldsymbol{u} \tag{14}$$

$$= (\cos\tau, \sin\tau, 0) \begin{pmatrix} -\cos\tau\,\sin\varphi - \cos\omega\,\cos\varphi\,\sin\tau \\ \cos\varphi\,\cos\tau - \cos\omega\,\sin\varphi\,\sin\tau \\ \sin\omega\,\sin\tau \end{pmatrix} \tag{15}$$

$$= \sin^2\left(\frac{\omega}{2}\right)\,\sin(2\tau) \tag{16}$$

A plot of the angular error $\cos^{-1}\big(\boldsymbol{d}\cdot\boldsymbol{u}\big)$ as a function of the angles ω and φ is shown in Figure 4(a), for $\tau+\varphi = 30°$. The plot in Figure 4(b) shows the angular error $\cos^{-1}\big(\boldsymbol{d}\cdot\boldsymbol{u}\big)$ as a funtion of the point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$ for the angle $\tau+\varphi = 30°$, and the function f given by (8).

2.3 Improving the Chen *et al.* Trackball

Let, the displacement vector $\boldsymbol{d}$ is in the image plane, and makes an angle $\tau+\varphi$ with the $\boldsymbol{x}-$axis

$$\boldsymbol{d} = \big(\cos(\tau+\varphi), \sin(\tau+\varphi), 0\big)^\top \tag{17}$$

In this section we will show that the rotation axis by the corrected Chen *et al.* approach is given by:

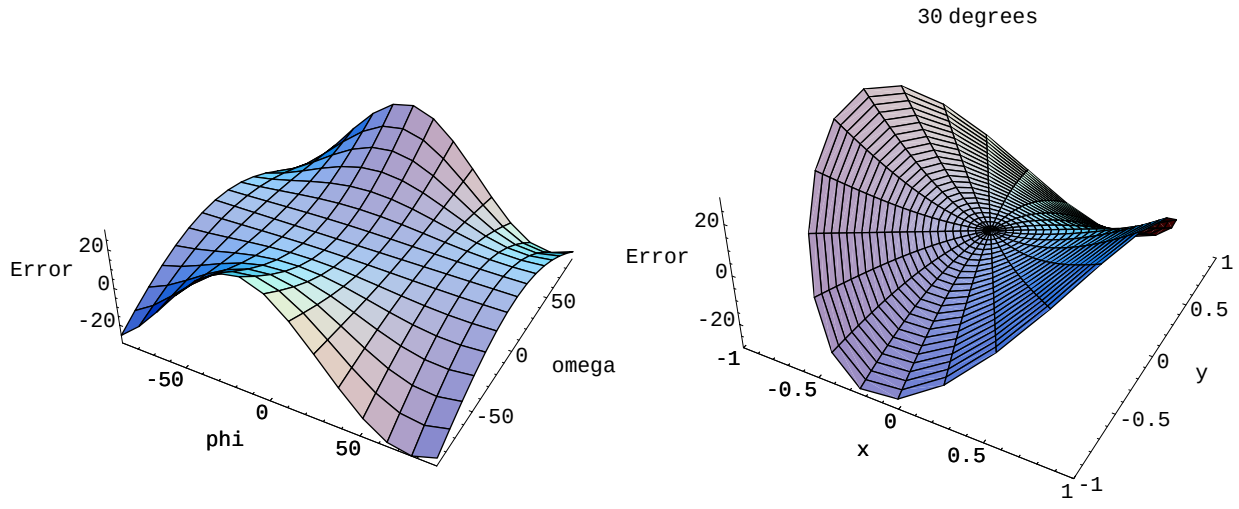
$$\boldsymbol{u} = \begin{pmatrix} -\cos\omega\,\sin(\tau+\varphi) \\ \cos\omega\,\cos(\tau+\varphi) \\ \sin\omega\,\sin\tau \end{pmatrix}. \tag{18}$$

The displacement vector $\boldsymbol{d}$ is located at point $\boldsymbol{p}_a$, see Figure 2. First, rotate the displacement vector $\boldsymbol{d}$ the angle $-\varphi$ around the $\boldsymbol{z}-$axis. The vector $\boldsymbol{d}$ is located at point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$, so the angle φ is equal to $\varphi = \tan^{-1}\big(y_a/x_a\big)$, and the transformed displacement vector $\widehat{\boldsymbol{d}}$ is equal to

$$\widehat{\boldsymbol{d}} = \begin{pmatrix} \cos(-\varphi) & -\sin(-\varphi) & 0 \\ \sin(-\varphi) & \cos(-\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\tau+\varphi) \\ \sin(\tau+\varphi) \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\tau \\ \sin\tau \\ 0 \end{pmatrix} \tag{19}$$

After this transformation, the displacement vector $\widehat{\boldsymbol{d}}$ is on the $\boldsymbol{x}-$axis, see Figure 3(b).

The second step is to transform the displacement vector $\widehat{\boldsymbol{d}}$ to the origin $\boldsymbol{o}$ by a rotation $-\omega$ around the $\boldsymbol{y}-$axis. Specifying the angle ω is deferred to later in this article.



(a) (b)

Figure 4: **(a)** The angular error $\cos^{-1}(\boldsymbol{d}\cdot\boldsymbol{u})$ (in degrees) between the displacement vector $\boldsymbol{d}$ and the transformed rotation axis $\boldsymbol{u}$, (14)—(16) as a function of the point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$. In this plot the function f is given by (8), and the angle $\tau + \varphi = 30°$.
(b) The angular error $\cos^{-1}(\boldsymbol{d}\cdot\boldsymbol{u})$ (in degrees) between the displacement vector $\boldsymbol{d}$ and the transformed rotation axis $\boldsymbol{u}$, (14)—(16) as a function of the point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$. In this plot the function f is given by (8), and the angle $\tau + \varphi = 30°$.

The original displacement vector $\boldsymbol{d}$ is transformed to the origin by the transformation

$$\widetilde{\boldsymbol{d}} = \begin{pmatrix} \cos(-\omega) & 0 & \sin(-\omega) \\ 0 & 1 & 0 \\ -\sin(-\omega) & 0 & \cos(-\omega) \end{pmatrix} \begin{pmatrix} \cos(-\varphi) & -\sin(-\varphi) & 0 \\ \sin(-\varphi) & \cos(-\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\tau+\varphi) \\ \sin(\tau+\varphi) \\ 0 \end{pmatrix} \tag{20}$$

$$= \begin{pmatrix} \cos(-\omega) & 0 & \sin(-\omega) \\ 0 & 1 & 0 \\ -\sin(-\omega) & 0 & \cos(-\omega) \end{pmatrix} \begin{pmatrix} \cos\tau \\ \sin\tau \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\omega\,\cos\tau \\ \sin\tau \\ \cos\tau\,\sin\omega \end{pmatrix} \tag{21}$$

After these two rotations, the displacement vector $\widetilde{\boldsymbol{d}}$ starts at the origin $\boldsymbol{o}$, see Figure 3(a). From (21) it is seen that the $z-$component is different from zero, i.e. it is *not* in the image plane, as opposed to the Chen *et al.* approach, see Section 2.1.

The unknown rotation axis $\widetilde{\boldsymbol{u}}$ should be perpendicular to the displacement vector. Now, the displacement vector has been transformed to the origin $\boldsymbol{o}$, i.e. the transformed displacement vector $\widetilde{\boldsymbol{d}}$ is given by (20). Therefore, the rotation axis might be computed as the cross-product of the $\boldsymbol{z}-$axis and $\widetilde{\boldsymbol{d}}$.

Given two vectors $\boldsymbol{a} = (a_x, a_y, a_z)^\top$ and $\boldsymbol{b} = (b_x, b_y, b_z)^\top$, the cross-product $\boldsymbol{a}\times\boldsymbol{b}$ may be written as follows

$$\boldsymbol{a}\times\boldsymbol{b} = \boldsymbol{M}(\boldsymbol{a})\,\boldsymbol{b} \tag{22}$$

where

$$\boldsymbol{M}(\boldsymbol{a}) = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{23}$$

The $\boldsymbol{z}-$axis has coordinates $(0,0,1)^\top$, so the matrix $\boldsymbol{M}\big((0,0,1)^\top\big)$ looks like this

$$\boldsymbol{M}\big((0,0,1)^\top\big) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{24}$$

Hence, the rotation axis $\widetilde{\boldsymbol{u}}$ can be written as the product of $\boldsymbol{M}\big((0,0,1)^\top\big)$, (24), and $\widetilde{\boldsymbol{d}}$, (21),

$$\widetilde{\boldsymbol{u}} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \cos\omega\, \cos\tau \\ \sin\tau \\ \cos\tau\, \sin\omega \end{pmatrix} = \begin{pmatrix} -\sin\tau \\ \cos\omega\, \cos\tau \\ 0 \end{pmatrix} \tag{25}$$

From (25) it can be seen that $\widetilde{\boldsymbol{u}}$ is in the image plane, but it has a different orientation than the rotation axis computed by Chen *et al.*, (2).

Now that the rotation axis $\widetilde{\boldsymbol{u}}$ has been computed as if the displacement vector was at the origin, it has to be transformed back to its original position $\boldsymbol{p}_a$. This is done by first rotating the axis by the angle ω around the $\boldsymbol{y}-$axis yielding the rotation axis $\widehat{\boldsymbol{u}}$

$$\widehat{\boldsymbol{u}} = \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & \cos\omega \end{pmatrix} \begin{pmatrix} -\sin\tau \\ \cos\omega\, \cos\tau \\ 0 \end{pmatrix} = \begin{pmatrix} -\cos\omega\, \sin\tau \\ \cos\omega\, \cos\tau \\ \sin\omega\, \sin\tau \end{pmatrix} \tag{26}$$

The rotation axis has been transformed as if the displacement vector was on the $\boldsymbol{x}-$axis, (26). The last thing to do is to transform that rotation axis to the point $\boldsymbol{p}_a$. This is done by a rotation by the angle φ around the $\boldsymbol{z}-$axis

$$\boldsymbol{u} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -\cos\omega\, \sin\tau \\ \cos\omega\, \cos\tau \\ \sin\omega\, \sin\tau \end{pmatrix} = \begin{pmatrix} -\cos\omega\, \sin(\tau+\varphi) \\ \cos\omega\, \cos(\tau+\varphi) \\ \sin\omega\, \sin\tau \end{pmatrix} \tag{27}$$

To see that this rotation axis $\boldsymbol{u}$ is always perpendicular to the displacement vector $\boldsymbol{d}$, one can compute the dot-product of the vectors $\boldsymbol{d}$ (17) and $\boldsymbol{u}$ (27)

$$\boldsymbol{d} \cdot \boldsymbol{u} = \boldsymbol{d}^\top \boldsymbol{u} = \big(\cos(\tau+\varphi), \sin(\tau+\varphi), 0\big) \begin{pmatrix} -\cos\omega\, \sin(\tau+\varphi) \\ \cos\omega\, \cos(\tau+\varphi) \\ \sin\omega\, \sin\tau \end{pmatrix} \tag{28}$$

$$= -\cos\omega\, \sin(\tau+\varphi)\cos(\tau+\varphi) + \cos\omega\, \cos(\tau+\varphi)\sin(\tau+\varphi) = 0 \tag{29}$$

The point $\boldsymbol{p}_a$ is assumed to be in the image plane, so its coordinates are $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$. Then the rotation angle around the $\boldsymbol{z}-$axis can be computed as

$$\varphi = \tan^{-1}\left(\frac{y_a}{x_a}\right) \tag{30}$$

The rotation angle around the $\boldsymbol{y}-$axis may be computed in several ways. In the paper by [1] it is computed as a function f of the distances $\mid \boldsymbol{p}_a \mid$ and $\mid \boldsymbol{r} \mid$ as follows

$$\omega = f\left(\frac{\mid \boldsymbol{p}_a \mid}{\mid \boldsymbol{r} \mid}\right) \tag{31}$$

where $f : \mathbb{R} \longrightarrow \mathbb{R}$ is a monotone function which satisfies

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \dfrac{\pi}{2} & \text{if} \quad x \geq 1 \end{cases} \tag{32}$$

At least three choices for the function f are relevant to discuss:

- Original suggestion by Chen *et al.*, see (8):

$$f(x) = x$$

- The angle actually specified by the user under perspective projection of the virtual sphere:

$$f(x) = \left(\frac{\left(r^2 - O_z^2\right)\sqrt{1-x^2} - O_z x^2 r}{O_z^2 - r^2\left(1-x^2\right)} \right)$$

  The proof is omitted due to page limitations.

- Shoemake's trackball, the angle specified by the user under orthographic projection of the virtual sphere.

$$f(x) = \sin^{-1}(x)$$

  A proof is given in Section 3.2.

3 Shoemake's Trackball

Shoemake [22] and others [23] implement a special version of the virtual trackball, the so-called arcball, by mapping the points in the image plane $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ onto a virtual sphere with radius $|\boldsymbol{r}|$ behind the screen by a function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$. The function $\boldsymbol{m}$ maps a point $\boldsymbol{p} = (x, y, 0)^\top; x, y \in [-1, 1]$ in the image plane onto a point $\boldsymbol{P}$ on the virtual sphere, such that $\boldsymbol{p}$ is the orthographic projection of $\boldsymbol{P}$, see Figure 1. The function $\boldsymbol{m}$ is given by

$$\boldsymbol{m}(\boldsymbol{p}) = \boldsymbol{m}(x, y, 0) = \boldsymbol{P} = \begin{cases} \left(x, y, \sqrt{r^2 - (x^2 + y^2)}\right)^\top & \text{if} \quad \sqrt{x^2 + y^2} \leq r \\ \dfrac{r}{\sqrt{x^2 + y^2}} (x, y, 0)^\top & \text{if} \quad \sqrt{x^2 + y^2} > r \end{cases} \tag{33}$$

The graph of the function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ which maps 3D screen coordinates $(x, y, 0)$ to 3D coordinates on the virtual sphere is shown in Figure 1. First the points $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ are mapped to the points $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ respectively. Then the sphere is rotated an angle θ along the great circular arc defined by the origin $\boldsymbol{O}$, $\boldsymbol{P}_a$, and $\boldsymbol{P}_c$. This rotation is performed by rotating the angle θ around an axis $\boldsymbol{u}$ which is perpendicular to both of the location vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$. The rotation axis $\boldsymbol{u}$ and the angle θ are given by

$$\boldsymbol{u} = \boldsymbol{P}_a \times \boldsymbol{P}_c \tag{34}$$

$$\theta = \tan^{-1}\left(\frac{|\, \boldsymbol{P}_a \times \boldsymbol{P}_c \,|}{\boldsymbol{P}_a \cdot \boldsymbol{P}_c} \right) \tag{35}$$

The graph of the function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ is shown in Figure 5(a). Shoemake uses quaternions to compute the rotation angle θ and axis $\boldsymbol{u}$. If the vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ are normalized they can be represented as unit quaternions $\boldsymbol{Q}_a$ and $\boldsymbol{Q}_c$

$$\boldsymbol{Q}_a = \left(0, \frac{\boldsymbol{P}_a}{|\, \boldsymbol{P}_a \,|}\right) \tag{36}$$

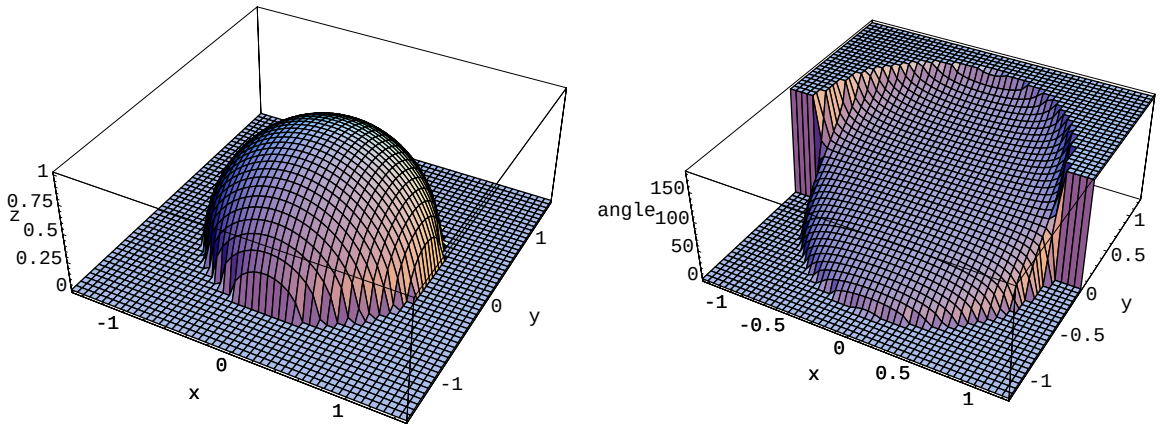
$$\boldsymbol{Q}_c = \left(0, \frac{\overline{\boldsymbol{OP}_c}}{|\, \overline{\boldsymbol{OP}_c} \,|}\right) \tag{37}$$

and the unit quaternion which rotates $\boldsymbol{P}_a$ into $\boldsymbol{P}_c$ along a great circle is given by [24, page 6]

$$\sqrt{-\boldsymbol{Q}_a \boldsymbol{Q}_c} \tag{38}$$

That is, $\boldsymbol{Q}_a$ is rotated into $\boldsymbol{Q}_c$ as follows

$$\boldsymbol{Q}_c = \left(\sqrt{-\boldsymbol{Q}_a \boldsymbol{Q}_c}\right) \boldsymbol{Q}_a \left(\sqrt{-\boldsymbol{Q}_a \boldsymbol{Q}_c}\right)^{-1} \tag{39}$$



(a) (b)

Figure 5: **(a)**The graph of the 3D points computed by Shoemake. **(b)** The angle between the $\boldsymbol{Z}-$axisand the axis of rotation for $\boldsymbol{p}_a = (-1.5, 0, 0)^\top$.

The product $-\boldsymbol{Q}_a\boldsymbol{Q}_c$ is a unit quaternion and so is its square root $\sqrt{-\boldsymbol{Q}_a\boldsymbol{Q}_c}$. Their relation is given by [24, page 6] as

$$-\boldsymbol{Q}_a\boldsymbol{Q}_c = (\cos\theta, \boldsymbol{n}\sin\theta) \quad \Longrightarrow \quad \sqrt{-\boldsymbol{Q}_a\boldsymbol{Q}_c} = \left(\cos\frac{\theta}{2}, \boldsymbol{n}\sin\frac{\theta}{2}\right) \tag{40}$$

where $\boldsymbol{n}$ is defined in the appendix. Shoemake does not compute $\sqrt{-\boldsymbol{Q}_a\boldsymbol{Q}_c}$, but for convenience he instead rotates by $\boldsymbol{Q}_a\boldsymbol{Q}_c^{-1}$ which is equivalent to $-\boldsymbol{Q}_a\boldsymbol{Q}_c$ because of the relations:

$$\boldsymbol{Q}_a\boldsymbol{Q}_c^{-1} = \boldsymbol{Q}_a\boldsymbol{Q}_c^{*} = -\boldsymbol{Q}_a\boldsymbol{Q}_c \tag{41}$$

It is seen that Shoemake computes a rotation of 2θ instead of θ.

3.1 Remarks on Shoemake's Approach

This section contains two remarks on Shoemake's approach to virtual trackballs. One remark is about the angle of rotation, the other one is about a discontinuity in the direction of the rotation axis.

The intention was that the trackball should rotate an angle θ, i.e. expressed as a quaternion, the rotation should be equal to $\boldsymbol{Q} = \left(\cos\frac{\theta}{2}, \boldsymbol{n}\sin\frac{\theta}{2}\right)$, but Shoemake actually computes the quaternion $\boldsymbol{Q} = (\cos\theta, \boldsymbol{n}\sin\theta)$, which represents a rotation of 2θ.

The intention was that the trackball should rotate objects continuously while the mouse was dragged over the screen. Unfortunately, the direction of the axis of rotation has a discontinuity, which will be shown in this section.

Figure 6 shows two situations where the mouse is clicked at a point $\boldsymbol{p}_a$ and dragged across the screen to a point $\boldsymbol{p}_c$ where both points are outside the projection of the virtual sphere. In both cases the points $\boldsymbol{p}_a, \boldsymbol{p}_c$ are mapped onto 3D points $\boldsymbol{P}_a, \boldsymbol{P}_c$ on the silhouette of the virtual sphere, and therefore both vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ are parallel to the image plane. The axis of rotation is equal to the crossproduct of the vectors, i.e. $\boldsymbol{P}_a \times \boldsymbol{P}_c$.

Figure 6(a) shows a situation, where the axis of rotation goes into the paper, and Figure 6(b) shows a situation, where the axis of rotation goes out of the paper. That is, the axes of rotation are perpendicular to the image plane. The actual rotations are show as thick great circular arcs in the figure. If the point $\boldsymbol{P}_c$ lies exactly on the dashed line, the axis of rotation is the zero vector.

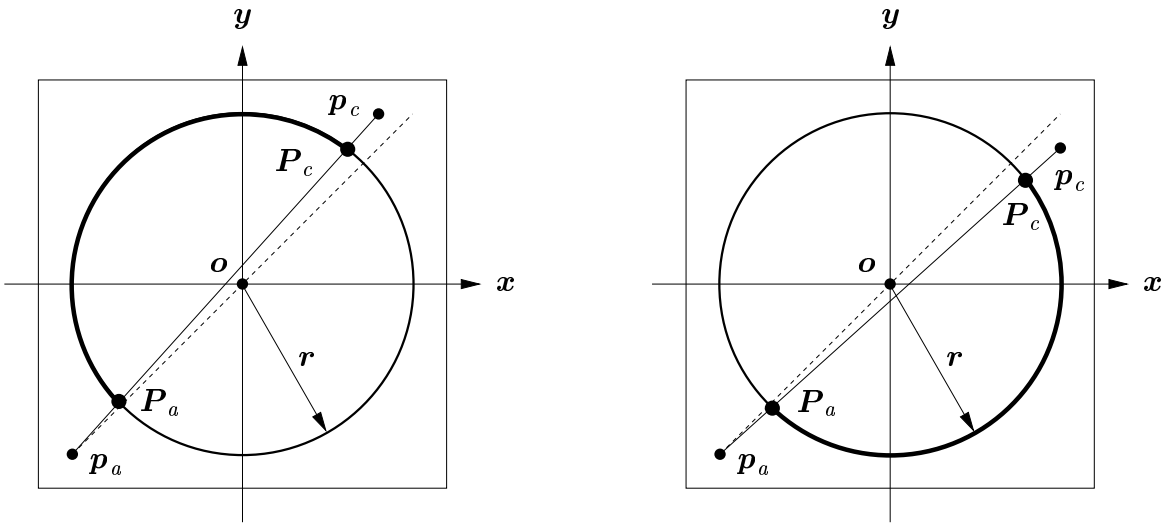


Figure 6: The mouse is clicked at point $\boldsymbol{p}_a$ and dragged across the screen to a point $\boldsymbol{p}_c$, where both points are outside the projection of the virtual sphere. **(a)** In this case the vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ are both parallel to the image plane, so the axis of rotation is directed into the paper, and the rotation is along the thick great circular arc. **(b)** In this case the vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ are both parallel to the image plane, so the axis of rotation is directed out of the paper, and the rotation is along the thick great circular arc. If the point $\boldsymbol{P}_c$ lies exactly on the dashed line, the axis of rotation is the zero vector.

If the point $\boldsymbol{p}_c$ moves outside of the projection of the virtual sphere just around the dashed line in Figure 6 it can be seen that the axis of rotation flips in and out of the paper each time the dashed line is crossed. As long as the point $\boldsymbol{p}_c$ is outside of the projection of the virtual sphere the user will not notice the discontinuity, because the actual rotation behaves as if it had been a 2D rotation, and rotations commute in 2D. The discontinuity becomes visible when the point $\boldsymbol{p}_c$ moves into the projection of the virtual sphere, because when $\boldsymbol{p}_c$ crosses the silhouette of the virtual sphere, the vector $\boldsymbol{P}_c$ is no longer parallel to the image plane, and the actual axis of rotation is not perpendicular to the image plane. Therefore, the great circular arcs of rotation will start to sweep across the virtual sphere, but the way they sweep depends on where the point $\boldsymbol{p}_c$ enters the projection of the virtual sphere. This behavior is very visible to the user, and sometimes it can look rather strange, because the vector $\boldsymbol{P}_c$ changes very rapidly when it crosses the silhouette of the virtual sphere.

3.2 Shoemake's Trackball is a Special Case of Chen *et al.*'s Trackball

This section shows that the virtual trackball described by Shoemake is a special case of the virtual trackball described by Chen *et al.*

Recall that Chen *et al.* rotate an angle ω around the $\boldsymbol{y}-$axis, where

$$\omega = f\left(\frac{|\,\boldsymbol{p}_a\,|}{|\,\boldsymbol{r}\,|}\right) \tag{42}$$

and $f : \mathbb{R} \longrightarrow \mathbb{R}$ is a monotone function which satisfies

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \dfrac{\pi}{2} & \text{if} \quad x \geq 1 \end{cases} \tag{43}$$

Shoemake uses orthographic projection to map points $\boldsymbol{p}_a$ in the image plane onto the virtual trackball as shown in Figure 7.

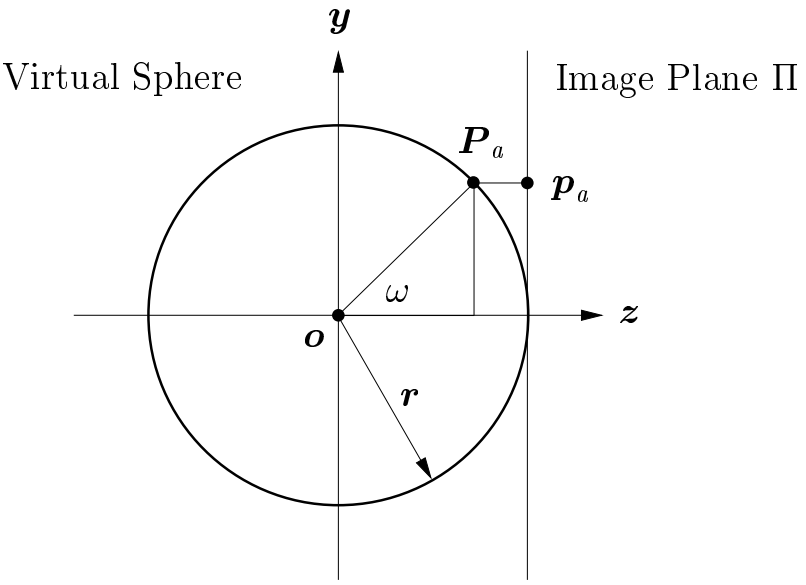


Figure 7: Orthographic projection of points $\boldsymbol{p}_a$ in the image plane onto the virtual sphere.

From the figure it can be seen that the ratio of $|\,\boldsymbol{p}_a\,|$ and $|\,\boldsymbol{r}\,|$ equals $\sin\omega$. That is, in this case

$$\omega = \sin^{-1}\left(\frac{|\,\boldsymbol{p}_a\,|}{|\,\boldsymbol{r}\,|}\right) = \sin^{-1}\left(\frac{\sqrt{x_a^2 + y_a^2}}{r}\right) \tag{44}$$

If the function f used by Chen *et al.* is chosen to be

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \sin^{-1}(x) & \text{if} \quad 0 \leq x \leq 1 \\ \dfrac{\pi}{2} & \text{if} \quad x \geq 1 \end{cases} \tag{45}$$

then the trackball described by Shoemake is a special case of the improved Chen *et al.* trackball described in Section 2.3.

4 Holroyd's Trackball

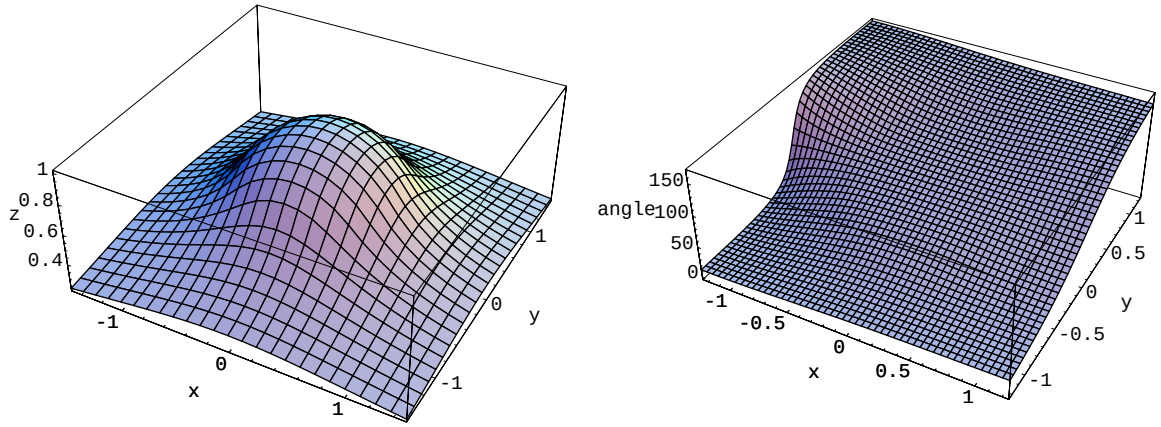
The virtual trackball implemented by Holroyd [20] is very similar to the trackball implemented by [22]. The difference between them is the way the points in the image plane $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ are mapped to 3D points. While Shoemake maps the points onto a sphere, Holroyd maps them onto a surface made by combining a sphere and a hyperbola: if the 2D point is close to the center of the screen the surface is a sphere, else it is a hyperbola. The function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ is given by

$$\boldsymbol{m}(\boldsymbol{p}) = \boldsymbol{m}(x, y, 0) = \boldsymbol{P} = \begin{cases} \left(x, y, \sqrt{r^2 - (x^2 + y^2)}\right)^\top & \text{if} \quad \sqrt{x^2 + y^2} \leq \dfrac{r}{\sqrt{2}} \\ \left(x, y, \dfrac{r^2}{2\sqrt{x^2 + y^2}}\right)^\top & \text{if} \quad \sqrt{x^2 + y^2} > \dfrac{r}{\sqrt{2}} \end{cases} \tag{46}$$

The graph of the function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ which maps 3D screen coordinates $(x, y, 0)$ to 3D coordinates on the virtual surface (46) is shown in Figure 8(a). The actual rotation is computed using the same equations as Shoemake, i.e. (36) — (39).

4.1 Remarks on Holroyd's Approach

In contrast to Shomake's trackball, Holroyd's trackball rotates more smoothly, because the orientation of the axis of rotation is continuous as a function of the screen coordinates. This is shown in Figure 8(b) where the angle between the $\boldsymbol{Z}-$axisand the axis of rotation is shown as a function of screen coordinates.



(a) (b)

Figure 8: **(a)** The graph of the 3D points computed by Holroyd. **(b)** The angle between the $\boldsymbol{Z}-$axisand the axis of rotation as a function of screen coordinates for $\boldsymbol{p}_a = (-1.5, 0, 0)^\top$.

5 Evaluations of the usability of virtual trackballs

To our knowledge, four studies have empirically investigated the usability of virtual trackballs. These studies are reviewed below. In addition, we point out some methodological limitations of the studies.

Chen *et al.* [1] report two experiments. Both these experiments required subjects to rotate a single object from an orientation chosen at random to a fixed position used throughout the experiment. After performing each rotation, subjects were given feedback on the accuracy of their rotation. In the first experiment, 12 subjects solved a series of nine tasks with each of the following techniques: Chen *et al.*'s virtual trackball; a controller-based technique using the mouse button to switch between $\boldsymbol{XY}$ and $\boldsymbol{Z}$ rotation; and two controller-based techniques using sliders. Chen *et al.*'s results indicate no practical difference in accuracy between the techniques. For tasks that require rotation around only one axis, the techniques using sliders were faster. However, for tasks requiring rotation around more than one axis, the virtual trackball and the $\boldsymbol{XY}+\boldsymbol{Z}$ technique were fastest. In the second experiment, six subjects used a virtual trackball and a controller-based technique where gestures determined which dimension to rotate on (as in [5]). This experiment did not reveal any difference between techniques. In both experiments, most subjects preferred the virtual trackball, stating that it felt more natural.

Jacob & Olivier [25] compared four techniques also used in Chen *et al.*'s experiment: a $\boldsymbol{XY}+\boldsymbol{Z}$ controller, the virtual trackball, Evans *et al.*'s [5] technique, and a controller-based technique with overlapping sliders. In addition to Chen *et al.*'s rotation task, Jacob & Olivier also had the 137 subjects perform an inspection task, which required subjects to examine an object to answer questions about it (for example, in a house find the number of windows). Subjects did 12 rotation and 6 inspection tasks with each controller. For the rotation task, the Evans *et al.* technique had higher mean errors compared to the other techniques. For task completion times, the virtual trackball was faster, the overlapping sliders and Evans *et al.*'s technique had comparable task completion times, and the $\boldsymbol{XY}$ and $\boldsymbol{Z}$ rotation were slowest. For the inspection task, the virtual trackball appears to be fastest; controllers have similar levels of error.

Hinckley *et al.* [19] compared Chen *et al.*'s [1] virtual trackball with Shoemake's [2] virtual trackball and two multiple-degree-of-freedom techniques. Twenty-four subjects used each technique to solve 10 tasks. The tasks required subjects to rotate an object from a fixed position to a randomly generated one. Hinckley *et al.* find no difference in accuracy between the techniques. However, the multiple-degrees-of-freedom techniques are between 33% and 36% faster than the virtual trackballs. Hinckley *et al.* consider their results to show that the faster movement of Shoemake's trackball does not decrease users' satisfaction. On the contrary, many users report that they liked the trackball's responsiveness. The main usability problem with the virtual trackballs is that users are unsure about the difference between being inside and outside the centre of the virtual trackballs.

Partala [26] had 12 subjects use a virtual trackball, a modified version of the virtual trackball called the virtual rectangle, and a keyboard. Subjects had to match the orientation of an object shown on the screen to an object shown on paper. These techniques were implemented using two different metaphors: a metaphor of rotating the object (world-in-hand metaphor; rotation is controlling object) and moving one self around the object (eyeball-in-hand; rotation is controlling viewport). Partala's results show that the virtual trackball and the virtual rectangle had similar task completion times. When the world-in-hand metaphor was used, both the virtual rectangle and the virtual trackball were faster than a keyboard. Subjective satisfaction indicated that the virtual rectangle was preferred over the virtual trackball, which was preferred over the keyboard.

In our view, the evaluations of usability reviewed above suffer from several methodological limitations. First, the tasks used in the experiment are mostly simple rotation tasks and lack any direct resemblance to actual work tasks. It is not evident that the results generalize to more complex tasks, where for example subjects are deeply concentrated on solving a diagnostic problem. One exception is the study by Jacob & Olivier [25] which includes an inspection task. Interestingly, this study finds differences between controllers for different tasks. The technique by Evans *et al.* does relatively worse on the inspection tasks and Jacob & Olivier [25] note that "The overlapping sliders [...] seem more adequate for orientation tasks [we call those rotation tasks] than for inspection ones", p. 75. Even though this inspection task is very simple, it shows the utility of varying task types.

Second, usability may be seen as comprised of the aspects efficiency (e.g. time), effectiveness (e.g. rotation accuracy), and subjective satisfaction [27]. The studies by Chen *et al.* [1] and Jacob & Olivier [25] fail to report subjective satisfaction in a systematic way, potentially leaving out important differences between controllers. Furthermore, the three usability aspects may be measured by a variety of indicators. In the studies reviewed above, we have few usability measures in addition to time and accuracy. How mentally demanding is it, for example, to rotate objects using the various techniques? What understanding of the objects do users build? Which controller results in the least physical fatigue? These questions seem important, but have not been studied.

Third, the studies reviewed all emphasize accuracy and give subjects feedback on accuracy. Consequently, we do not now much about what happens if subjects are encouraged to emphasize speed (which have been done in other contexts, see [28, 13]).

Fourth, the studies leave out important detail about the implementation of the virtual trackballs making comparison difficult. In this article we have shown that Shoemake's trackball is discontinuous when the user presses the mouse outside the projection of the virtual sphere, making the size of the sphere crucial for usability studies. Lack of details such as size of the virtual sphere, how the sphere is visually indicated on the screen, etc, makes comparisons across studies difficult. More importantly, designers are not well equipped to making decisions about how to implement virtual trackballs when the details of the techniques used in the studies are not clear.

In summary, the evaluations of the usability of virtual trackballs are illuminating, yet seriously flawed. The evaluations suggest that accuracy of virtual trackballs are comparable to that of other techniques for rotating 3D objects. As found by Hinckley *et al.* [19], however, multiple-degrees-of-freedom devices are more efficient. Such devices also lead to higher subjective satisfaction. The distinction between inside and outside the virtual sphere is disturbing to users. To our knowledge no studies has been performed investigating the usability of Holroyd's solution. Nevertheless, users find the higher control to display ratio of Shoemake's trackball useful. However, the studies have serious methodological limitations; further studies using a wider range of tasks and richer measures of usability are needed.

6 Conclusion

Virtual trackballs are a convenient tool for rotation of 3D objects with a 2D mouse as a simulation of a physical trackball. Most often, the virtual trackballs are not displayed on screen, but are simulated as if situated in the center of the object of interest and have a size proportional to the objects spatial extend.

To our knowledge, Chen *et al.* [1] pioneered the field, while the method by Shoemake [2] and Holroyd [20] are the most commonly used. In this article the method by Chen *et al.* has been thoroughly discussed, corrected, and improved, and in the process we proved that Shoemake is an implementation of the corrected Chen *et al.* method.

In contrast to physical trackballs, virtual trackballs may often be used outside the range of projection, i.e. while you cannot operate a physical trackball without actually touching it, virtual trackballs may be operated outside their projection on the screen. From a usability point of view, this seems natural, since the virtual trackball is often not displayed and the user does not have any notion of the simulated, physical trackball. Chen *et al.* do not consider this possibility. Shoemake does, and this article has presented a hitherto neglected analysis of the Shoemake extension, and as a result it has been demonstrated that for certain mouse movements, the Shoemake extension is disturbingly discontinuous. Holroyd presents a solution, of which the consequences for usability is unknown.

Virtual trackballs are used for almost all rotations of 3D objects with a 2D mouse. This work has revealed and corrected a shaky mathematical foundation of virtual trackballs. Second, most usability experiments reported in the literature lack crucial detail for us to be able to evaluate the consequences of the corrections. Finally, these experiments use an unrealistic task: rotating a house to a pre-specified angle, making such experiments of limited value for judging the usability of virtual trackballs for real day-to-day work with 3D objects.

References

[1] Michael Chen, S.Joy Mountford, and Abigail Sellen, "A study in interactive 3-d rotation using 2-d control devices," *Computer Graphics*, vol. 22, no. 4, pp. 121–129, Aug. 1988.

[2] Ken Shoemake, "Arcball: A user interface for specifying three-dimensional orientation using a mouse," in *Proceedings of Graphics Interface'92*, 1992, pp. 151–156.

[3] "`http://www.3dmax.com`," .

[4] Stuart Card, Jock Mackinlay, and Ben Shneiderman, *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers, Inc, San Fransisco, USA, 1999.

[5] Kenneth B. Evans, Peter P. Tanner, and Marceli Wein, "Tablet based valuators that provides one, two or three degrees of freedom," in *Proceedings of SIGGRAPH 81*, 1981, In Computer Graphics, 15:3, pp. 91–97.

[6] Jef Raskin, *The Humane Interface: New Directions for Designing Interactive Systems*, Addison Wesley, Reading, MA, 2000.

[7] Ben Shneiderman, "Direct manipulation: A step beyond programming languages," *IEEE Computer*, vol. 16, no. 8, pp. 57–68, 1983.

[8] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill, "A hand gesture interface," in *Proceedings of CHI+GI 1987*, 1987, pp. 189–192.

[9] "`http://www.immersion.com`," .

[10] Shumin Zhai, Paul Milgram, and William Buxton, "The influence of muscle groups on performance of multiple degree-of-freedom input," in *Proceedings of CHI '96*, 1996, pp. 308–315.

[11] "`http://www.logicad3d.com`," .

[12] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell, "Passive real-world interface props for neurosurgical visualization," in *Proceedings of the ACM CHI 94*, 1994, pp. 452–458.

[13] Colin Ware and Jeff Rose, "Rotating virtual objects with real handles," *ACM Transactions on Computer-Human Interactions*, vol. 6, no. 2, pp. 162–180, 1999.

[14] "`http://www.vrweb.com`," .

[15] Lars Bretzner and Tony Lindeberg, "Use your hand as a 3-D mouse or relative orientation from extended sequences of sparse point and line correspondences using the affine trifocal tensor," in *Proceedings of the European Conference on Computer Vision*, 1998, Lecture Notes in Computer Science, 1406, pp. 141–157, Springer Verlag, Berlin, June 1998.

[16] W. Buxton and B. Myers, "A study in two-handed input," in *Proceedings of CHI*, 1986, pp. 321–326.

[17] Lawrence D. Cutler, Bernd Fr olich, and Pat Hanrahan, "Two-handed direct manipulation on the responsive workbench," in *Symposium on Interactive 3D Techniques*, 1997.

[18] M.W. Gribnau, I.M. Verstijnen, and J. M. Hennessey, "Three dimensional object orientation using the non-dominant hand," in *4th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*, 1998.

[19] Ken Hinckley, Joe Tullio, Randi Pausch, Dennis Proffitt, and Neal Kassell, "Usability analysis of 3D rotation techniques," in *Proceedings of UIST 97*, 1997, pp. 1–10.

[20] Tom Holroyd, "Holroyd's trackball," `http://members.tripod.com/professor_tom/`.

[21] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley Publishing Company, Reading, Massachesetts, 2nd edition, 1996.

[22] Ken Shoemake, "Arcball rotation control," vol. IV of *Graphics Gems*, chapter III, pp. 175–192. Academic Press, Inc., 1994.

[23] Jeff Hultquist, "A virtual trackball," vol. I of *Graphics Gems*, chapter 9, pp. 462–463. Academic Press, Inc., 1990.

[24] Edward Pervin and Jon A. Webb, "Quaternions in computer vision and robotics," Tech. Rep. CMU-CS-82-150, Department of Computer Science, Carnegie-Mellon University, 1982.

[25] Inés Jacob and Javier Oliver, "Evaluation of techniques for specifying 3D rotations with a 2D input device," in *Proceedings of HCI 95*, 1995, pp. 63–76.

[26] Timo Partala, "Controlling a single 3D object: viewpoint metaphors, speed and subjective satisfaction," in *Human-Computer Interaction – INTERACT 99*, 1999, pp. 536–543.

[27] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk, "Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated?," in *Proceedings of CHI 2000*, 2000, pp. 345–352.

[28] Colin Ware, "Using hand position for virtual object placement," *Visual Computer*, vol. 6, pp. 245–253, 1990.