Virtual Trackballs Revisited

Knud Henriksen, Jon Sporring, and Kasper Hornbæk

Abstract—Rotation of three-dimensional objects by a two-dimensional mouse is a typical task in 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. This article reviews and provides a mathematical foundation for virtual trackballs. The first, but still popular, virtual trackball was described by Chen et al. [1]. We show that the virtual trackball by Chen et al. does not rotate the object along the intended great circular arc on the virtual trackball and we give a correction. Another popular virtual trackball is Shoemake's quaternion implementation [2], which we show to be a special case of the virtual trackball by Chen et al.. Shoemake extends the scope of the virtual trackball to the full screen. Unfortunately, Shoemake's virtual trackball is inhomogeneous and discontinuous with consequences for usability. Finally, we review Bell's virtual trackball [3] and discuss studies of the usability of virtual trackballs.

Index Terms—Virtual trackball, arcball, 3D rotation, 2D mouse, mathematical foundation, usability review.

1 Rotating 3D Objects

Rotation of 3D objects is crucial in software for computer-aided design, operation simulations, and desktop virtual reality. This paper reviews and provides a mathematical foundation for one technique for rotating objects, the virtual trackball. In addition, we discuss how the usability of virtual trackballs has been evaluated.

User interface techniques for rotating 3D objects are most often of the following four kinds: In *view-based techniques*, several views of the object to be rotated are presented to the user. In each view, the user may rotate the object on one or two dimensions using a controller, such as a slider. One common implementation is to present three views of the object, corresponding to the xy, xz, and yz projections. View-based techniques are used in commercial applications [4] and research prototypes [1]. The drawbacks of view-based techniques are that the views take up screen real-estate, that rotation is normally only possible on one dimension at a time, and that the user may experience problems in mentally integrating the different views of the object (as found in interfaces presenting both an overview and a detail view of an information space [5, p. 634]).

In *controller-based techniques*, each dimension the object can be rotated on is manipulated with a controller [6]. Such controllers may be represented separately in the user interface as sliders or buttons on the keyboard [1], may overlap the object to be rotated, or may be invisible and activated through gestures, e.g., a circular motion of the mouse rotates the object around the Z-axis [7]. The user may also use buttons to choose what dimension a certain control rotates on. For example, pressing and holding down the left mouse button may switch from rotating on the X and Y dimensions to rotating on the Z-dimension [7]. The drawbacks of controller-based techniques are that they only allow rotation on one dimension, that the dimensions to be rotated are sometimes fixed to the initial orientations of the coordinate system, and that switching between controllers takes time or depends on the user interface being in a particular mode, which is a known cause of user difficulties [8].

In *virtual trackball techniques*, rotation is controlled with a projection of mouse movement onto a virtual trackball, which in turn controls the actual rotation of the object [1], [2], [4]. Virtual trackballs allow rotation along several dimensions simultaneously and integrate controller and the object controlled, as in direct manipulation [9]. The main drawback of virtual trackballs is a lack of thorough mathematical description of the projection from mouse movement onto a rotation.

In *multiple-degree-of-freedom techniques*, input devices with more than two degrees of freedom are used for rotating objects. Some of these techniques track translation and rotation by sensors embedded in devices such as gloves [10], [11], circular ball-like objects [12], [13], real world objects such as dolls or handles [14], [15], joysticks [16], and so-called 3D mice [13]. Using computer vision techniques, the gestures of a person's hand have also been used to rotate objects [17]. The main drawback of multiple-degree-of-freedom techniques is the need for special devices; however, these techniques may be more efficient for rotating 3D objects than other techniques [18], [15].

In addition to these four kinds, other techniques for rotating 3D objects exist. Two-handed interaction [19] with 3D objects is one promising way of interaction where users control some dimensions of rotation with one hand and other dimensions with the mouse [20], [21].

This paper reviews and extends the virtual trackball technique for rotating a 3D object with a 2D mouse. We focus on the virtual trackball for three reasons. First, even though multiple-degree-of-freedom techniques are promising in terms of usability [18], [15], an ordinary 2D mouse is

- *K. Henriksen is with the Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark. E-mail: kaiip@diku.dk.*
- *J. Sporring is with the 3DLab, School of Dentistry, University of Copenhagen, Noerre Alle 20, DK-2200 Copenhagen, Denmark. E-mail: sporring@lab3d.odont.ku.dk.*
- *K. Hornbæk is with NIK, H.C. Oersted Institute, University of Copenhagan, Universitetsparken 5, DK-2100 Copenhagen, Denmark. E-mail: KHornbaek@nik.ku.dk.*

Manuscript received 16 July 2002; revised 19 May 2003; accepted 16 June 2003.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 116971.

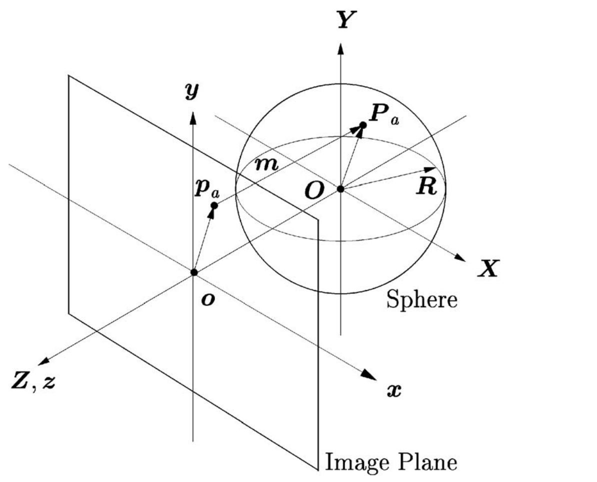


Fig. 1. A 2D point on the image plane is mapped to a 3D point on a sphere which is located behind the image plane.

likely to remain the main interaction device for a large user group. Second, the virtual trackball has established itself as an industry standard for rotating 3D objects with a 2D mouse. For example, it is used in commercial applications ranging from Matlab [22] to Maya [6]. Third, as mentioned above, the virtual trackball has a number of useful features compared to other techniques that use a 2D mouse. In particular, it shares screen real-estate with the objects it controls and it allows several dimensions to be rotated at the same time.

In Sections 2, 3, 4, and 5, we review and extend common techniques for implementing virtual trackballs. Section 6 discusses usability problems with the four kinds of user interface techniques and reviews experimental evaluations of the techniques. Section 7 presents our conclusions.

2 Virtual Trackballs

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

The basic mathematical framework is as follows: The 2D screen is embedded into a 3D image plane with its own coordinate system xyz. A screen coordinate $\boldsymbol{p}_a = (x_a, y_a)^\top$ is thought of as a 3D point $\boldsymbol{p}_a = (x_a, y_a, 0)^\top$ in the image plane. The mapping from the image plane to the 3D sphere is specified by a function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$. The function $\boldsymbol{m}$ is typically chosen to be the orthographic projection. The motion of the mouse on the screen may thus be projected as motion of the 3D sphere.

To determine the rotation from the mouse movement, assume that the mouse is 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. The corresponding points on the sphere are $\boldsymbol{P}_a = \boldsymbol{m}(\boldsymbol{p}_a)$ and $\boldsymbol{P}_c = \boldsymbol{m}(\boldsymbol{p}_c)$, which together with $\boldsymbol{O}$ define a great circular arc. This arc is chosen as the rotation, see Fig. 2.

Let us consider the possible rotations that may be specified by a virtual trackball. Lie algebra [23], [24] is a well-founded mathematical tool for studying the possible

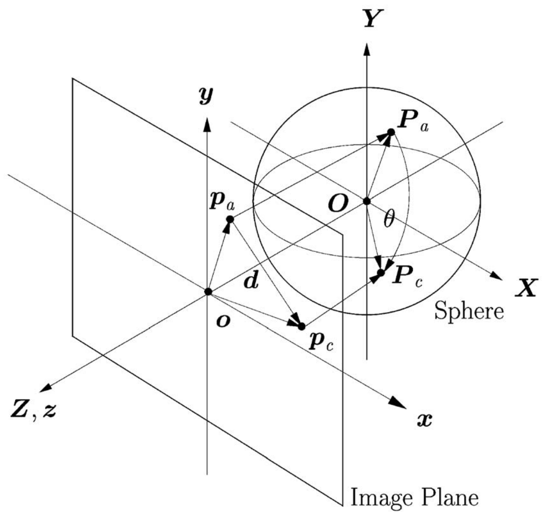


Fig. 2. A virtual trackball can be thought of as a 3D sphere located behind the screen. The points $\boldsymbol{p}$ on the image plane are mapped to points $\boldsymbol{P}$ on the 3D sphere by a function $\boldsymbol{m} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$, i.e., $\boldsymbol{P} = \boldsymbol{m}(\boldsymbol{p})$.

mappings between motions in the image plane onto rotational motion; however, we will take a more pictorial approach: Once the mouse has been pressed at the point $\boldsymbol{p}_a$, the possible axes of rotation are given by all the great circles passing through $\boldsymbol{P}_a = \boldsymbol{m}(\boldsymbol{p}_a)$. Pictorially, $\boldsymbol{P}_a$ is the north pole and the great circles are the longitudinal lines of the sphere. A great circle is selected by releasing the mouse at point $\boldsymbol{P}_c = \boldsymbol{m}(\boldsymbol{p}_c)$. Hence, if $\boldsymbol{p}_a$ is set on a point on the projected sphere (including its rim), then the realizable axes of rotation are all perpendicular to the line joining $\boldsymbol{P}_a$ with $\boldsymbol{O}$. With orthographic projection of the sphere, the closest half of the sphere, the hemisphere, and its projection on the screen are in one-to-one relation and any point on the closest hemisphere may therefore be selected by the user. Hence, any axis of rotation may be specified by two points $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ on the orthographically projected sphere since all great circles extend onto a given hemisphere. This is not the case for perspective projection of the sphere since less than half the sphere is visible on the screen. Axes of rotation that may be obtained are limited to lying outside a cone around the z-axis, where the size of the cone is proportional to the distance to the sphere divided by the focal length.

The amount of rotation is specified by the length of the great circular arc between $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$. It would appear that rotation around the z-axis under orthographic projection is limited to be either positive or negative. However, the sign of rotation may be flipped by interchanging the positions of $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ on the rim of the projected sphere. The length of the great circular arc is largest when $\boldsymbol{p}_c$ is set on the rim such that the line joining $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ passes through $\boldsymbol{o}$. The length of the great circular arc is smallest when $\boldsymbol{p}_c$ is set on the exact opposite side of the rim of the projected sphere.

In the following, we discuss three popular virtual trackballs specifying the above projection: Chen et al. [1], Shoemake [2], and Bell [3].

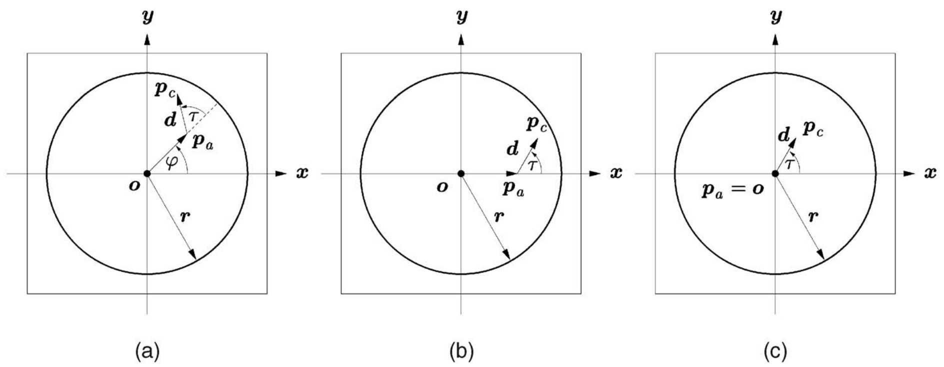


Fig. 3. Screen view where the mouse is pressed at point p_a and moved to point p_c where it is released. (a) The general case (Chen et al. case 3). (b) The point p_a is located on the x-axis (Chen et al. case 2). (c) The point p_a is located at the origin, i.e., $p_a = o = (0,0,0)^\top$ (Chen et al. case 1).

3 The Chen et al. Virtual Trackball

This section gives a review of the virtual trackball by Chen et al. [1]. In Fig. 3a, the circular projection of the sphere and two user selected points inside the projection are shown: $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$. The location vector $\boldsymbol{p}_a$ makes an angle φ with the $\boldsymbol{x}$-axis and the displacement vector $\boldsymbol{d}$ makes the angle $\tau + \varphi$ with the $\boldsymbol{x}$-axis,

$$\boldsymbol{d} = \boldsymbol{p}_c - \boldsymbol{p}_a = d(\cos(\tau+\varphi), \sin(\tau+\varphi), 0)^\top. \tag{1}$$

The scalar $d = |\boldsymbol{d}|$ is the Euclidean length of the displacement vector.

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.

3.1 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 Fig. 3c. 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 (1) has $\varphi = 0$. 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. 3b. 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. In this special case, the unit axis of rotation $\widehat{\boldsymbol{u}}$ will be

$$\widehat{\boldsymbol{u}} = \boldsymbol{R}_y(\omega)\widetilde{\boldsymbol{u}}, \tag{3}$$

where $\boldsymbol{R}_y(\omega)$ is a matrix representing a rotation around the $\boldsymbol{y}$-axis

$$\boldsymbol{R}_y(\omega) = \begin{pmatrix} \cos\omega & 0 & \sin\omega \\ 0 & 1 & 0 \\ -\sin\omega & 0 & cos\omega \end{pmatrix}. \tag{4}$$

Unfortunately, this is incorrect as it turns out that $\widehat{\boldsymbol{u}}$ is not necessarily perpendicular to the displacement vector $\boldsymbol{d}$ (see Sections 3.2 and 3.3 for a detailed discussion). Before discussing the correction, we finish the review of the virtual trackball of Chen et al. in order to introduce the remaining important concepts.

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 Fig. 3a. The actual unit rotation axis can be obtained by rotating the axis, $\widehat{\boldsymbol{u}}$, the angle φ around the $\boldsymbol{z}$-axis. The final unit rotation axis $\boldsymbol{u}$ becomes

$$\boldsymbol{u} = \boldsymbol{R}_z(\varphi)\widehat{\boldsymbol{u}} = \boldsymbol{R}_z(\varphi)\boldsymbol{R}_y(\omega)\widetilde{\boldsymbol{u}} \tag{5}$$

$$= \begin{pmatrix} -\cos\tau\,\sin\varphi - \cos\omega\,\cos\varphi\,\sin\tau \\ \cos\varphi\,\cos\tau - \cos\omega\,\sin\varphi\,\sin\tau \\ \sin\omega\,\sin\tau \end{pmatrix}, \tag{6}$$

where $\boldsymbol{R}_z(\varphi)$ is a matrix representing a rotation around the $\boldsymbol{z}$-axis

$$\boldsymbol{R}_z(\varphi) = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

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

$$\varphi = \tan^{-1}\left(\frac{y_a}{x_a}\right). \tag{8}$$

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

$$\omega = f\left(\frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}\right). \tag{9}$$

The function f is monotone and satisfies

$$f(x) = \begin{cases} 0 & \text{if} \quad x \le 0 \\ \frac{\pi}{2} & \text{if} \quad x \ge 1. \end{cases} \tag{10}$$

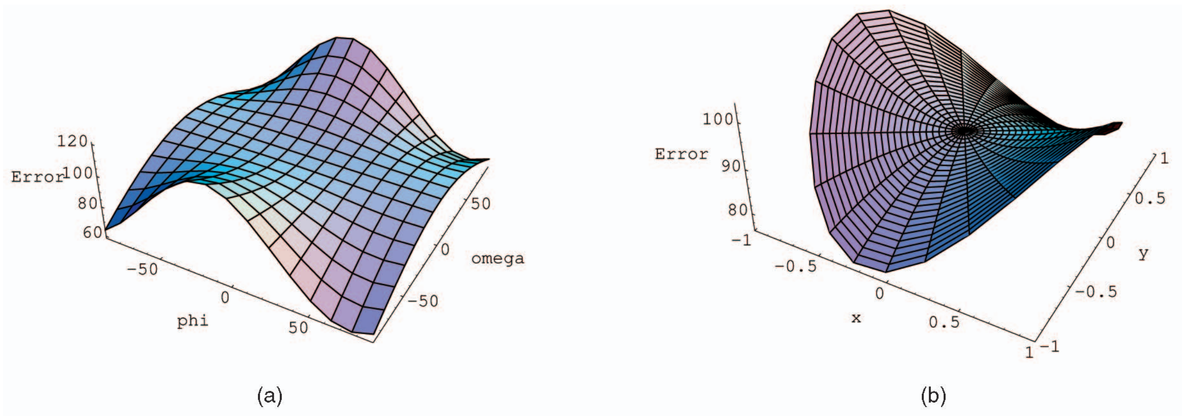


Fig. 4. The angular error $\cos^{-1}(\frac{\boldsymbol{d}}{|\boldsymbol{d}|}\cdot\boldsymbol{u})$ in degrees between the displacement vector and the transformed rotation axis as a function of $\boldsymbol{p}_a=(x_a,y_a,0)^\top$. Here, f is given by (11) and $\tau+\varphi=30^\circ$. Graph (a) shows the error as a function of φ and ω and (b) illustrates the same error as a function of x and y.

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

$$f(x)=\begin{cases}0 & \text{if} \quad x\le 0\\ \frac{\pi}{2}x & \text{if} \quad 0\le x\le 1\\ \frac{\pi}{2} & \text{if} \quad x\ge 1.\end{cases} \tag{11}$$

The actual rotation matrix, which rotates a point the angle θ (shown in Fig. 2) around the axis $\boldsymbol{u}=(u_x,u_y,u_z)^\top$, is given by Foley et al. [25, p. 227].

In the paper by Chen et al. [1] the angle θ is chosen by experiment to be

$$\theta=\frac{\pi}{2}\frac{|\boldsymbol{d}|}{|\boldsymbol{r}|}\left(1-\left(1-\frac{0.2}{\pi}\right)\frac{2\omega}{\pi}(1-|\cos\tau|)\right), \tag{12}$$

where $\boldsymbol{d}$ is given by (1), ω is given by (9)-(11), and τ is as shown in Fig. 3a.

3.2 Remarks on the Chen et al. Virtual Trackball

Unfortunately, the rotation axis $\boldsymbol{u}$ computed by (5) is incorrect because it is not necessarily perpendicular to the displacement vector (1). Using $\tau+\varphi$ as the angle between the displacement vector and the $\boldsymbol{x}$-axis, the rotation axis (5) is found to be

$$\boldsymbol{u}=\begin{pmatrix}-\cos\tau\,\sin\varphi-\cos\omega\,\cos\varphi\,\sin\tau\\ \cos\varphi\,\cos\tau-\cos\omega\,\sin\varphi\,\sin\tau\\ \sin\omega\,\sin\tau\end{pmatrix}. \tag{13}$$

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

$$\frac{\boldsymbol{d}}{|\boldsymbol{d}|}\cdot\boldsymbol{u}=\frac{\boldsymbol{d}}{|\boldsymbol{d}|}^\top\boldsymbol{u}=-\sin^2\left(\frac{\omega}{2}\right)\sin(2\tau). \tag{14}$$

Fig. 4a shows a plot of the angular error $\cos^{-1}(\frac{\boldsymbol{d}}{|\boldsymbol{d}|}\cdot\boldsymbol{u})$ as a function of the angles ω and φ for $\tau+\varphi=30^\circ$. The plot in Fig. 4b shows the angular error $\cos^{-1}(\frac{\boldsymbol{d}}{|\boldsymbol{d}|}\cdot\boldsymbol{u})$ as a function of the point $\boldsymbol{p}_a=(x_a,y_a,0)^\top$ for the angle $\tau+\varphi=30^\circ$ and the function f (11).

3.3 Improving the Chen et al. Virtual Trackball

In this section, we will show that the rotation axis by the corrected Chen et al. virtual trackball is given by:

$$\boldsymbol{u}=\begin{pmatrix}-\cos\omega\,\sin(\tau+\varphi)\\ \cos\omega\,\cos(\tau+\varphi)\\ \sin\omega\,\sin\tau\end{pmatrix}. \tag{15}$$

Initially, the displacement vector $\boldsymbol{d}$ is located at point $\boldsymbol{p}_a$, see Fig. 3a. To transform it to $\boldsymbol{o}$, first we rotate it by the angle $-\varphi$ around the $\boldsymbol{z}$-axis. The angle φ is equal to $\varphi=\tan^{-1}(y_a/x_a)$, since $\boldsymbol{p}_a=(x_a,y_a,0)^\top$. The transformed displacement vector $\widehat{\boldsymbol{d}}$ is therefore equal to

$$\widehat{\boldsymbol{d}}=\boldsymbol{R}_z(-\varphi)\boldsymbol{d}=d(\cos\tau,\sin\tau,0)^\top. \tag{16}$$

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

Second, 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 Section 3.4.

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

$$\widetilde{\boldsymbol{d}}=\boldsymbol{R}_y(-\omega)\boldsymbol{R}_z(-\varphi)\boldsymbol{d} \tag{17}$$

$$=d(\cos\omega\,\cos\tau,\sin\tau,\cos\tau\,\sin\omega)^\top. \tag{18}$$

After these two rotations, the displacement vector $\widetilde{\boldsymbol{d}}$ starts at the origin $\boldsymbol{o}$, see Fig. 3c. It is emphasized that the z-component in (18) is different from zero and therefore *not* in the image plane. This is in contrast to (1) used by the Chen et al. virtual trackball.

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 (18). 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:

$$\boldsymbol{a}\times\boldsymbol{b}=\boldsymbol{M}(\boldsymbol{a})\,\boldsymbol{b}, \tag{19}$$

where

$$\boldsymbol{M}(\boldsymbol{a})=\begin{pmatrix}0 & -a_z & a_y\\ a_z & 0 & -a_x\\ -a_y & a_x & 0\end{pmatrix}. \tag{20}$$

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

$$\boldsymbol{M}\big((0,0,1)^\top\big) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (21)$$

Hence, the rotation axis $\widetilde{\boldsymbol{u}}$ can be written as the product of $\boldsymbol{M}\big((0,0,1)^\top\big)$, (21), and $\widetilde{\boldsymbol{d}}$, (18),

$$\widetilde{\boldsymbol{u}} = \boldsymbol{M}\big((0,0,1)^\top\big)\widetilde{d} = \begin{pmatrix} -\sin\tau \\ \cos\omega\, \cos\tau \\ 0 \end{pmatrix}. \quad (22)$$

The vector $\widetilde{\boldsymbol{u}}$ is in the image plane with a different orientation than the rotation axis computed by Chen et al., where $\widetilde{\boldsymbol{u}} = (-\sin\tau, \cos\tau, 0)^\top$ by (2).

Because the rotation axis $\widetilde{\boldsymbol{u}}$ has been computed as if the displacement vector was at the origin, $\widetilde{\boldsymbol{u}}$ 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 $\widehat{\boldsymbol{u}}$:

$$\widehat{\boldsymbol{u}} = \boldsymbol{R}_y(\omega)\widetilde{\boldsymbol{u}} \quad (23)$$

$$= \big(-\cos\omega\, \sin\tau, \cos\omega\, \cos\tau, \sin\omega\, \sin\tau\big)^\top. \quad (24)$$

The rotation axis has been transformed as if the displacement vector was on the $\boldsymbol{x}$-axis using (23). Last, we transform the rotation axis to the point $\boldsymbol{p}_a$ by rotation φ around the $\boldsymbol{z}$-axis

$$\boldsymbol{u} = \boldsymbol{R}_z(\varphi)\widehat{\boldsymbol{u}} = \begin{pmatrix} -\cos\omega\, \sin(\tau+\varphi) \\ \cos\omega\, \cos(\tau+\varphi) \\ \sin\omega\, \sin\tau \end{pmatrix}. \quad (25)$$

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 $\boldsymbol{d}$ and $\boldsymbol{u}$:

$$\boldsymbol{d}\cdot\boldsymbol{u} = \begin{pmatrix} \cos(\tau+\varphi) \\ \sin(\tau+\varphi) \\ 0 \end{pmatrix} \cdot \begin{pmatrix} -\cos\omega\, \sin(\tau+\varphi) \\ \cos\omega\, \cos(\tau+\varphi) \\ \sin\omega\, \sin\tau \end{pmatrix} = 0. \quad (26)$$

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

$$\varphi = \tan^{-1}\left(\frac{y_a}{x_a}\right). \quad (27)$$

3.4 Choosing Function f

The rotation angle around the $\boldsymbol{y}$-axis may be computed in several ways. Chen et al. [1] computed it as a function f of the distances $|\boldsymbol{p}_a|$ and $|\boldsymbol{r}|$ as

$$\omega = f\left(\frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}\right), \quad (28)$$

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

$$f(x) = \begin{cases} 0 & \text{if} \quad x \le 0 \\ \frac{\pi}{2} & \text{if} \quad x \ge 1. \end{cases} \quad (29)$$

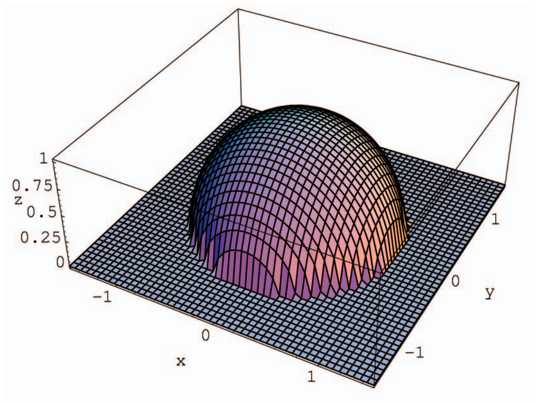


Fig. 5. The graph of the 3D points computed by Shoemake.

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

- The original suggestion by Chen et al., see (11):

$$f(x) = x.$$

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

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

  The derivation is given in Appendix A.
- The angle specified by the user under orthographic projection of the sphere:

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

  This choice results in the virtual trackball of Shoemake [2], to be demonstrated in Section 4.2.

4 The Shoemake Virtual Trackball

Shoemake [26] and Hultquist [27] implement a special version of the virtual trackball, the so-called arcball, in which $\boldsymbol{p}$ is the orthographic projection of $\boldsymbol{P}$ as shown in Fig. 2. The function $\boldsymbol{m}_{Shoemake}$ is extended to the full image by

$$m_{Shoemake}(\boldsymbol{p}) = m_{Shoemake}(x,y,0) = \boldsymbol{P}$$
$$= \begin{cases} \begin{pmatrix} x \\ y \\ \sqrt{r^2 - (x^2+y^2)} \end{pmatrix} & \text{if} \quad \sqrt{x^2+y^2} \le r \\ \frac{r}{\sqrt{x^2+y^2}}\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} & \text{if} \quad \sqrt{x^2+y^2} > r \end{cases} \quad (30)$$

such that a point outside the projected sphere is mapped to the nearest point on the rim of the sphere. The graph of the z-value of $m_{Shoemake}$ is shown in Fig. 5. In the virtual trackball of Shoemake, first, the point $\boldsymbol{p}_a$ is mapped to $\boldsymbol{P}_a$ and $\boldsymbol{p}_c$ to $\boldsymbol{P}_c$. 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

$$\boldsymbol{u} = \boldsymbol{P}_a \times \boldsymbol{P}_c \tag{31}$$

$$\theta = \tan^{-1}\left(\frac{|\boldsymbol{P}_a \times \boldsymbol{P}_c|}{\boldsymbol{P}_a \cdot \boldsymbol{P}_c}\right). \tag{32}$$

Shoemake uses quaternions to compute the rotation angle θ and the rotation axis $\boldsymbol{u}$; see Dam et al. [28] for a review of quaternions. If the vectors $\boldsymbol{P}_a$ and $\boldsymbol{P}_c$ are normalized, they can be represented as unit quaternions $\boldsymbol{Q}_a$ and $\boldsymbol{Q}_c$:

$$\boldsymbol{Q}_a = \left(0, \frac{\boldsymbol{P}_a}{|\boldsymbol{P}_a|}\right) \tag{33}$$

$$\boldsymbol{Q}_c = \left(0, \frac{\overline{\boldsymbol{P}_c}}{|\overline{\boldsymbol{P}_c}|}\right). \tag{34}$$

The unit quaternion which rotates $\boldsymbol{P}_a$ into $\boldsymbol{P}_c$ along a great circle is given by Pervin and Webb [29, p. 6]

$$\sqrt{-\boldsymbol{Q}_c\boldsymbol{Q}_a}. \tag{35}$$

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

$$\boldsymbol{Q}_c = \left(\sqrt{-\boldsymbol{Q}_c\boldsymbol{Q}_a}\right)\boldsymbol{Q}_a\left(\sqrt{-\boldsymbol{Q}_c\boldsymbol{Q}_a}\right)^{-1}. \tag{36}$$

The product $-\boldsymbol{Q}_c\boldsymbol{Q}_a$ is a unit quaternion and so is its square root $\sqrt{-\boldsymbol{Q}_c\boldsymbol{Q}_a}$. Pervin and Webb {29, p. 6] give their relation as

$$\begin{aligned} &-\boldsymbol{Q}_c\boldsymbol{Q}_a = (\cos\theta, \boldsymbol{n}\sin\theta) \\ &\Longrightarrow \sqrt{-\boldsymbol{Q}_c\boldsymbol{Q}_a} = \left(\cos\frac{\theta}{2}, \boldsymbol{n}\sin\frac{\theta}{2}\right), \end{aligned} \tag{37}$$

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

$$\boldsymbol{Q}_c\boldsymbol{Q}_a{}^{-1} = \boldsymbol{Q}_c\boldsymbol{Q}_a{}^{*} = -\boldsymbol{Q}_c\boldsymbol{Q}_a. \tag{38}$$

Thus, Shoemake computes a rotation of 2θ instead of θ.

4.1 Remarks on the Shoemake Virtual Trackball

We note that Shoemake uses the quaternion $\boldsymbol{Q} = (\cos\theta, \boldsymbol{n}\sin\theta)$, which implements a 2θ rotation. We will not judge whether this happened by accident or design, but we note that the quaternion implementing a rotation by θ is given by $\boldsymbol{Q} = (\cos\frac{\theta}{2}, \boldsymbol{n}\sin\frac{\theta}{2})$. In Section 6, we discuss some questions related to the usability of the virtual trackball of Shoemake.

The Shoemake trackball allows the user to rotate an object using any point on the screen by mapping points outside the projected sphere onto its rim (30). Unfortunately, the chosen implementation has a discontinuity which will be discussed in the following.

Fig. 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 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 rim of the sphere. 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 $\boldsymbol{P}_a \times \boldsymbol{P}_c$.

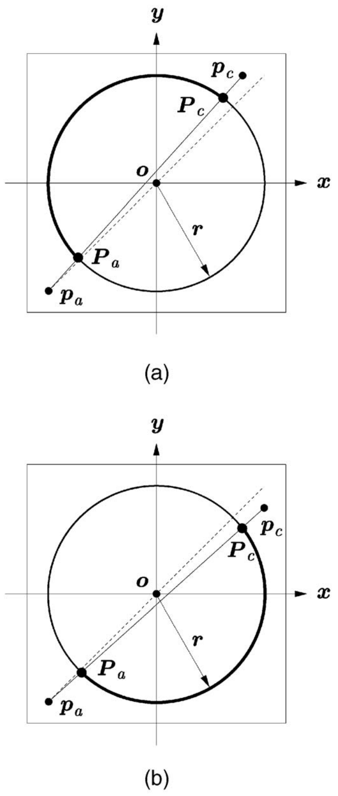
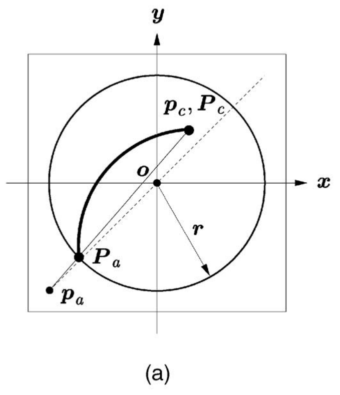


Fig. 6. The mouse is clicked at point p_a and dragged across the screen to a point p_c, where both points are outside the projection of the sphere. Vectors P_a and P_c are parallel to the image plane and the rotation is along the thick great circular arc with the rotation axis directed into the paper (a) and out of the paper (b). If the point P_c lies exactly on the dashed line, the axis of rotation is the zero vector.

Fig. 6a shows a situation where the axis of rotation goes into the paper and, in Fig. 6b, 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 shown 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.

If the point $\boldsymbol{p}_c$ moves outside of the projection of the sphere just around the dashed line in Fig. 6, the axis of rotation will flip in and out of the paper each time the dashed line is crossed. As long as the point $\boldsymbol{p}_c$ is outside the projection of the sphere, the user is not likely to notice the discontinuity since the actual rotation behaves as a 2D rotation. The discontinuity becomes visible when the point $\boldsymbol{p}_c$ moves into the projection of the sphere because, when $\boldsymbol{p}_c$ crosses the rim of the 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 sphere, but the way they sweep depends on where the point $\boldsymbol{p}_c$ enters the projection of the sphere, see Fig. 7. This behavior is visible to the user and looks rather strange because the vector $\boldsymbol{P}_c$ changes very rapidly when it crosses the rim of the sphere. Fig. 8 shows the angle between the $\boldsymbol{Z}$-axis and the rotation axis when $\boldsymbol{p}_a = (-1.5, 0, 0)^\top$ and the point $\boldsymbol{p}_c$ is moved across the screen parallel to the $\boldsymbol{x}$-axis.



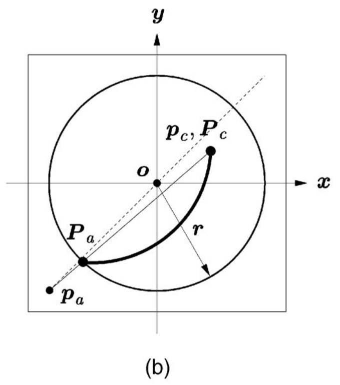


Fig. 7. The mouse is clicked at point p_a and dragged across the screen to a point p_c above (a) and below (b) the discontinuity line.

4.2 The Shoemake Virtual Trackball Is a Special Case of the Chen et al. Virtual 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

$$\omega = f\left(\frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}\right) \tag{39}$$

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

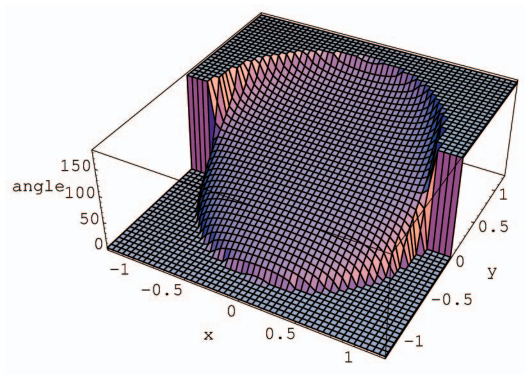


Fig. 8. The angle between the $\boldsymbol{Z}$-axis and the axis of rotation for $p_a = (-1.5, 0, 0)^\top$.

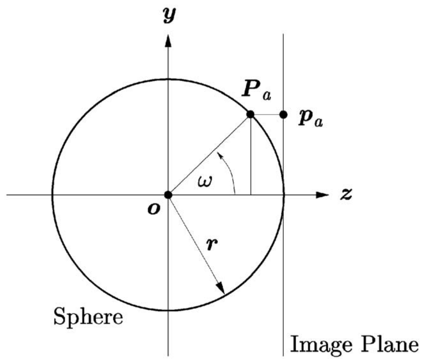


Fig. 9. Orthographic projection of point p_a in the image plane onto P_a on the sphere.

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \frac{\pi}{2} & \text{if} \quad x \geq 1. \end{cases} \tag{40}$$

Shoemake uses orthographic projection to map points $\boldsymbol{p}_a$ in the image plane onto the sphere, see Fig. 9. The figure shows that the ratio of $|\boldsymbol{p}_a|$ and $|\boldsymbol{r}|$ equals $\sin\omega$, i.e.,

$$\omega = \sin^{-1}\left(\frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}\right) = \sin^{-1}\left(\frac{\sqrt{x_a^2 + y_a^2}}{r}\right). \tag{41}$$

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

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ \sin^{-1}(x) & \text{if} \quad 0 \leq x \leq 1 \\ \frac{\pi}{2} & \text{if} \quad x \geq 1, \end{cases} \tag{28}$$

then the virtual trackball described by Shoemake is a special case of the improved Chen et al. virtual trackball described in Section 3.3.

5 The Bell Virtual Trackball

The virtual trackball implemented by Bell [3] is very similar to the virtual trackball implemented by Shoemake [26]. The difference is the function mapping points in the image plane to 3D points. While Shoemake maps the points onto a sphere, Bell 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 $\boldsymbol{m}_{Bell} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ is given by

$$\boldsymbol{m}_{Bell}(\boldsymbol{p}) = \boldsymbol{m}_{Bell}(x, y, 0) = \boldsymbol{P}$$
$$= \begin{cases} \begin{pmatrix} x \\ y \\ \sqrt{r^2 - (x^2 + y^2)} \end{pmatrix} & \text{if} \quad \sqrt{x^2 + y^2} \leq \frac{r}{\sqrt{2}} \\ \begin{pmatrix} x \\ y \\ \frac{r^2}{2\sqrt{x^2+y^2}} \end{pmatrix} & \text{if} \quad \sqrt{x^2 + y^2} > \frac{r}{\sqrt{2}}. \end{cases} \tag{43}$$

The graph of the function $\boldsymbol{m}$ is shown in Fig. 10. The actual rotation is computed using the same equations as Shoemake, i.e., (33)-(36).

5.1 Remarks on the Bell Virtual Trackball

In contrast to the Shoemake virtual trackball, the Bell virtual trackball rotates smoothly because the orientation of the

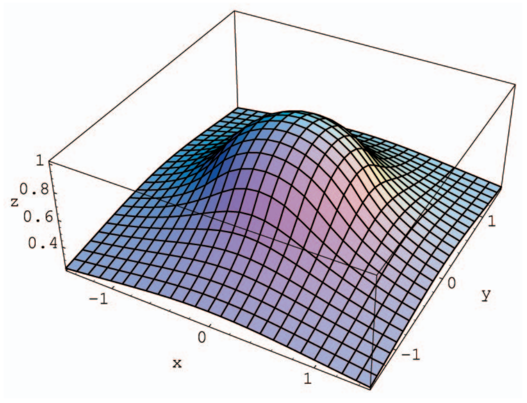


Fig. 10. The graph of the 3D points computed by Bell.

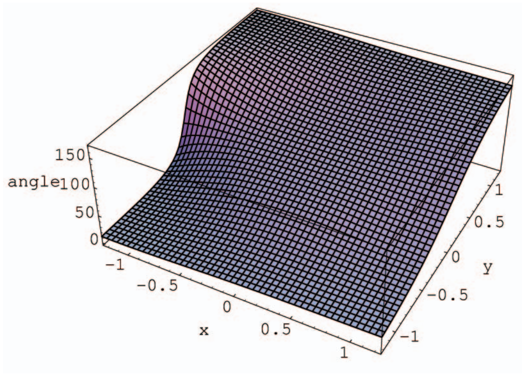


Fig. 11. The angle between the $\boldsymbol{Z}$-axis and the axis of rotation as a function of screen coordinates for $\boldsymbol{p}_a = (-1.5, 0, 0)^\top$.

axis of rotation is continuous as a function of the screen coordinates. Fig. 11 shows the angle between the $\boldsymbol{Z}$-axis and the axis of rotation as a function of screen coordinates and is seen to be smooth.

6 Evaluations of the Usability of Virtual Trackballs

To our knowledge, four studies have empirically evaluated the usability of virtual trackballs. These studies are reviewed below. In addition, we identify possibilities for future work in evaluating virtual trackballs.

Chen et al. [1] report two experiments. Both 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 nine tasks with each of the following techniques: the virtual trackball of Chen et al., a controller-based technique using the mouse button to switch between $\boldsymbol{XY}$ and $\boldsymbol{Z}$ rotation, and two controller-based techniques using sliders. The results of Chen et al. indicated no practical difference in accuracy between the techniques. For tasks that required rotation around only one axis, the techniques using sliders were faster. However, for tasks that require 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 [7]. 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 and Olivier [30] compared four techniques also used in the experiments of Chen et al.: an $\boldsymbol{XY} + \boldsymbol{Z}$ controller-based technique, the virtual trackball, the technique of Evans et al. [7], and a controller-based technique with overlapping sliders. In addition to the rotation task used by Chen et al., Jacob and Olivier 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 six inspection tasks with each controller. For the rotation task, the technique of Evans et al. resulted in a higher mean error compared to the other techniques. With respect to task completion times, the virtual trackball was faster, the overlapping sliders and the technique by Evans et al. had comparable task completion times, and the $\boldsymbol{XY} + \boldsymbol{Z}$ technique was slowest. For the inspection task, the virtual trackball appeared to be fastest; the four techniques had similar levels of error.

Hinckley et al. [18] compared the virtual trackball by Chen et al. with the virtual trackball of Shoemake 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. found no difference in accuracy between the techniques. However, the multiple-degrees-of-freedom techniques were between 33 percent and 36 percent faster than the virtual trackballs. Hinckley et al. consider their results to show that the faster movement of the virtual trackball of Shoemake does not decrease users' satisfaction. On the contrary, many users reported that they liked the virtual trackball's responsiveness [18, p. 7]. The main usability problem with the virtual trackballs was that users were unsure about the difference between being inside and outside the center of the virtual trackballs.

Partala [31] had 12 subjects use a virtual trackball, a modified version of the virtual trackball, called the virtual rectangle, and a keyboard. These techniques were implemented using two different metaphors: a metaphor of rotating the object (world-in-hand metaphor; rotation is controlling the object) and moving one's self around the object (eyeball-in-hand; rotation is controlling the viewport). Subjects had to match the orientation of an object shown on the screen to an object shown on paper. The results showed that the virtual trackball and the virtual rectangle have 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 to the virtual trackball, which, in turn, was preferred to the keyboard.

In our view, the evaluations of usability reviewed above raise questions to be explored in future work so as to reach a broader description of the usability of virtual trackballs. First, the tasks used in the experiments are mostly simple rotation tasks that require subjects to rotate an object from one position to another. One exception is the study by Jacob and Olivier [30] that includes an inspection task. Interestingly, this study finds differences between controllers for rotation and inspection tasks. The technique by Evans et al. does relatively worse on the inspection tasks, and Jacob and Olivier [30, p. 75] note that "The overlapping sliders [...] seem more adequate for orientation tasks [we call those rotation tasks] than for inspection ones." Consequently, using a broader selection of tasks in evaluations of virtual trackballs may show trade offs between different techniques. Future experiments could explore 1) a series of rotation tasks which may prove harder when the virtual trackball has discontinuities; 2) tasks closer to actual work tasks, for example, where subjects are concentrated on solving a diagnostic problem (such tasks may help explore how usable virtual trackballs are when attention has to be divided between controlling rotation and solving the work task); and 3) tasks similar to the informal painting task used by Kurtenbach and Balakrishnan [32], which may help focus on different aspects of usability (e.g., fun or engagement) than do the, for example, diagnostic tasks. The taxonomy of tasks described by Plaisant et al. [33], though developed for image browsing, may help identify additional kinds of tasks that could supplement the simple rotation tasks.

Second, usability may be seen as comprised of the aspects effectiveness (e.g., rotation accuracy), efficiency (e.g., time), and subjective satisfaction (e.g., preference) [34]. The studies by Chen et al. [1] and Jacob and Olivier [30] do not 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, few usability measures are employed 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 rotated do users build? Which technique results in the least physical fatigue? These questions seem important to address in future work.

Third, the studies reviewed all emphasize accuracy and give subjects feedback on accuracy. Consequently, we need to explore what happens if subjects are encouraged to emphasize speed, which has been done in other contexts [35], [15].

Fourth, details in the implementation of virtual trackballs are likely to have a large impact on usability. In this paper, we have shown that the virtual trackball of Shoemake 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. Unfortunately, the studies reviewed omit details about the size of the virtual sphere, how the sphere is visually indicated on the screen, etc., which makes comparisons across studies difficult. Future work could explore, for example, the influence on usability of the size of the virtual sphere more systematically so as to help designers make decisions about how to implement virtual trackballs. Another line of work could explore how different sensitivities of virtual trackballs influence usability and if nonisomorphic rotational mappings [36] work for virtual trackballs.

In summary, the evaluations suggest that the accuracy of virtual trackballs is comparable to that of other techniques for rotating 3D objects. As found by Hinckley et al. [18], however, multiple-degrees-of-freedom techniques are more efficient. Such techniques also lead to higher subjective satisfaction. The distinction between inside and outside the virtual sphere seems the single most critical usability problem. Yet, to our knowledge, no studies have investigated the usability of the solution of Bell, which in part address this problem. Finally, we suggest supplementing the studies reviewed with further work using a wider range of tasks and richer measures of usability.

7 Conclusion

Virtual trackballs are convenient tools for rotation of 3D objects with a 2D mouse that work by simulating a physical trackball. Most often, virtual trackballs are not displayed on screen, but are simulated as if situated at the center of the object to be rotated and having a size proportional to the object's size.

To our knowledge, Chen et al. [1] pioneered the field, while the methods by Shoemake [2] and Bell [3] are the most commonly used. In this paper, the method by Chen et al. has been discussed and corrected; in the process, we demonstrated that Shoemake's virtual trackball [2] is a special case of the corrected Chen et al. virtual trackball.

While a physical trackball cannot be operated without actually touching it, virtual trackballs may be used outside the range of projection. From a usability point of view, this possibility seems natural because the virtual trackball is often not shown and the user does not have any notion of the simulated, physical trackball. Chen et al. do not consider this possibility. Shoemake does and we demonstrate that, for certain mouse movements, the extension of Shoemake is discontinuous. Bell [3] suggests an alternative solution, but, to our knownledge, the consequences for usability have not yet been studied.

Virtual trackballs are used for almost all rotations of 3D objects with a 2D mouse. This paper has contributed a solid mathematical foundation for virtual trackballs. In addition, we have identified possibilities for future work concerning the evaluation of virtual trackballs, especially how to find new tasks and usability measures for such evaluations.

Appendix A
Choosing $\omega = f(k) = \gamma$ in Chen et al.

In the special case where the $\boldsymbol{p}_a$ is on the $\boldsymbol{x}$-axis, Chen et al. Case 2, a sigmoid function $f(|\boldsymbol{p}_a|/|\boldsymbol{r}|)$ is used to rotate around the y-axis as depicted in Fig. 12. We wish to find the function f such that

$$\omega = f\left(\frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}\right) = \gamma. \qquad (44)$$

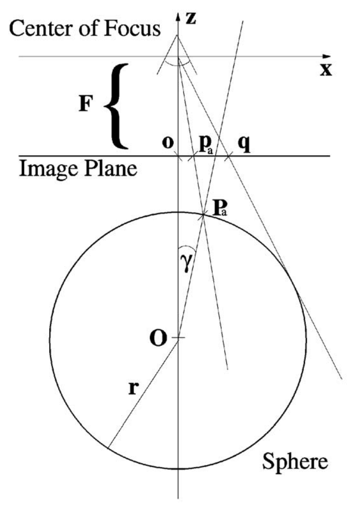


Fig. 12. A sphere centered on $z = 0$. The user specified point $\boldsymbol{p}_a$ projects onto the point P_a under perspective projection and the size of the projected sphere is determined by r.

In this section, we will show that

$$f(k) = \cos^{-1}\left(\frac{(r^2 - O_z^2)\sqrt{1-k^2} - O_z k^2 r}{O_z^2 - r^2(1-k^2)}\right) \tag{45}$$

using $k = \frac{|\boldsymbol{p}_a|}{|\boldsymbol{r}|}$ and independently of the focal length F.

In general, we have

$$\gamma = \cos^{-1}\left(\frac{(\boldsymbol{o} - \boldsymbol{O})\cdot(\boldsymbol{P}_a - \boldsymbol{O}))}{r|\boldsymbol{o} - \boldsymbol{O}|}\right) \tag{46}$$

with $\boldsymbol{o}$ at the center of focus. Because the sphere is assumed to be centered on the $\boldsymbol{z}$-axis and because $\boldsymbol{P}_a$ previously has been rotated around $\boldsymbol{O}$ such that $P_y = 0$, we may obtain a simplification of (46) using

$$r_y = O_x = O_y = 0 \tag{47}$$

such that

$$\omega = f\left(\frac{p_x}{r_x}\right) = \cos^{-1}\left(\frac{P_z - O_z}{r}\right). \tag{48}$$

Given a point $\boldsymbol{p}_a = (p_x, 0, 0)^\top$ in the image plane, the projection onto the sphere is the intersection of the projection line

$$l_p(s) = s\left(\boldsymbol{p}_a - \begin{pmatrix}0\\0\\F\end{pmatrix}\right) = s\begin{pmatrix}p_x\\0\\-F\end{pmatrix}, \tag{49}$$

with the implicit equation of the sphere:

$$(x - O_x)^2 + (y - O_y)^2 + (z - O_z)^2 - r^2 = 0. \tag{50}$$

The resulting second order polynomial,

$$(sp_x)^2 + (-sF - O_z)^2 - r^2 = 0, \tag{51}$$

is solved for s and the solution corresponding to the first intersection is selected

$$s' = \frac{-FO_z + \sqrt{(F^2 + p_x^2)r^2 - O_z^2 p_x^2}}{F^2 + p_x^2}. \tag{52}$$

Hence,

$$\gamma = \cos^{-1}\left(\frac{-s'F - O_z}{r}\right). \tag{53}$$

To find the sigmoid function f solving (44), (53) must be written in terms of p_x/q_x. To find q_x, a line is placed along $\boldsymbol{r}$

$$l_q(t) = t\left(\boldsymbol{q} - \begin{pmatrix}0\\0\\F\end{pmatrix}\right) = \begin{pmatrix}q_x\\0\\-F\end{pmatrix} \tag{54}$$

such that it grazes the sphere. Inserting $l_q(t)$ into the implicit equation for the sphere (50)

$$(tq_x)^2 + (-tF - O_z)^2 - r^2 = 0, \tag{55}$$

it is noted that $l_q(t)$ grazes the sphere when the second order polynomial has excactly one solution, i.e.,

$$(2O_zF)^2 = 4(F^2 + q_x^2)(O_z^2 - r^2). \tag{56}$$

The positive solution is

$$q_x = \frac{Fr}{\sqrt{O_z^2 - r^2}}. \tag{57}$$

Introducing the factor $k = p_x/q_x$ with $p_x = kq_x$, it is found that the sigmoid f such that $\omega = f(p_x/q_x) = \gamma$ is given by

$$\begin{aligned}\omega &= f(k)\\ &= \cos^{-1}\left(\frac{-s'F - O_z}{r}\right)\\ &= \cos^{-1}\left(\frac{(r^2 - O_z^2)\sqrt{1-k^2} - O_z k^2 r}{O_z^2 - r^2(1-k^2)}\right)\\ &= \gamma.\end{aligned} \tag{58}$$

Note that f is independent of the focal length F.

Appendix B

Reference Implementations

Implementations of the three virtual trackballs are available in C++ at: http://www.diku.dk/forskning/image/trackballs/.

References

[1] M. Chen, S.J. Mountford, and A. 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] K. Shoemake, "Arcball: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse," *Proc. Graphics Interface '92,* pp. 151-156, 1992.

[3] G. Bell, "Bell's Trackball," http://members.tripod.com/professor_tom/index.html, 1988.

[4] "3D Studio Max," http://www.3dmax.com, 2003.

[5] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think.* San Francisco: Morgan Kaufmann, 1999.

[6] "Maya from Alias/Wavefront," http://www.aliaswavefront.com, 2003.

[7] K.B. Evans, P.P. Tanner, and M. Wein, "Tablet Based Valuators that Provide One, Two or Three Degrees of Freedom," *Proc. SIGGRAPH '81, Computer Graphics,* vol. 15, no. 3, pp. 91-97, 1981.

[8] J. Raskin, *The Humane Interface: New Directions for Designing Interactive Systems.* Reading, Mass.: Addison Wesley, 2000.

[9] B. Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," *Computer,* vol. 16, no. 8, pp. 57-68, Aug. 1983.

[10] T.G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill, "A Hand Gesture Interface," *Proc. CHI+GI 1987,* pp. 189-192, 1987.

[11] "Cyberglove," http://www.immersion.com, 2003.

[12] S. Zhai, P. Milgram, and W. Buxton, "The Influence of Muscle Groups on Performance of Multiple Degree-of-Freedom Input," *Proc. Computer Human Interaction Conf. '96,* pp. 308-315, 1996.

[13] "Logitech," http://www.logicad3d.com, 2003.

[14] K. Hinckley, R. Pausch, J.C. Goble, and N.F. Kassell, "Passive Real-World Interface Props for Neurosurgical Visualization," *Proc. ACM Computer Human Interaction Conf. '94,* pp. 452-458, 1994.

[15] C. Ware and J. Rose, "Rotating Virtual Objects with Real Handles," *ACM Trans. Computer-Human Interactions,* vol. 6, no. 2, pp. 162-180, 1999.

[16] "Spacestick," http://www.vrweb.com, 2003.

[17] L. Bretzner and T. 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," *Proc. European Conf. Computer Vision,* pp. 141-157, June 1998.

[18] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell, "Usability Analysis of 3D Rotation Techniques," *Proc. User Interface Software and Techonology Symp. '97,* pp. 1-10, 1997.

[19] W. Buxton and B. Myers, "A Study in Two-Handed Input," *Proc. Computer Human Interaction,* pp. 321-326, 1986.

[20] L.D. Cutler, B. Frölich, and P. Hanrahan, "Two-Handed Direct Manipulation on the Responsive Workbench," *Proc. Symp. Interactive 3D Techniques,* 1997.

[21] M.W. Gribnau, I.M. Verstijnen, and J.M. Hennessey, "Three Dimensional Object Orientation Using the Non-Dominant Hand," *Proc. Fourth Int'l Conf. Design and Decision Support Systems in Architecture and Urban Planning,* 1998.

[22] "Matlab," http://www.mathworks.com, 2003.

[23] H. Flanders, *Differential Forms with Applications to the Physical Sciences.* Dover, 1989.

[24] F.W. Warner, *Foundations of Differential Manifolds and Lie Groups.* Springer, 1983.

[25] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice,* second ed. Addison-Wesley, 1996.

[26] K. Shoemake, "Arcball Rotation Control," *Graphics Gems,* P.S. Heckbert, ed., vol. IV, pp. 175-192, Academic Press, 1994.

[27] J. Hultquist, "A Virtual Trackball," *Graphics Gems,* A.S. Glassner, ed. , vol. I, chapter 9, pp. 462-463, Academic Press, 1990.

[28] E.B. Dam, M. Koch, and M. Lilholm, "Quaternions, Interpolation and Animation," Technical Report DIKU-98-5, Dept. of Computer Science, Univ. of Copenhagen, Denmark, July 1998.

[29] E. Pervin and J.A. Webb, "Quaternions in Computer Vision and Robotics," Technical Report CMU-CS-82-150, Dept. of Computer Science, Carnegie-Mellon Univ., 1982.

[30] I. Jacob and J. Oliver, "Evaluation of Techniques for Specifying 3D Rotations with a 2D Input Device," *Proc. Human Computer Interaction Symp. '95,* pp. 63-76, 1995.

[31] T. Partala, "Controlling a Single 3D Object: Viewpoint Metaphors, Speed and Subjective satisfaction," *Proc. Human-Computer Interaction—INTERACT '99,* pp. 536-543, 1999.

[32] G. Kurtenbach and R. Balakrishnan, "Exploring Bimanual Camera Control and Object Manipulation in 3D Graphics Interfaces," *Proc. Computer Human Interaction '99,* pp. 56-63, 1999.

[33] C. Plaisant, D. Carr, and B. Shneiderman, "Image Browsers: Taxonomy, Guidelines, and Informal Specifications," *IEEE Software,* vol. 12, no. 2, pp. 21-32, 1995.

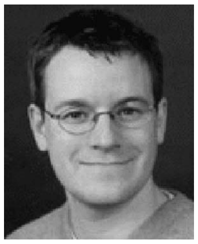
[34] E. Frøkjær, M. Hertzum, and K. Hornbæk, "Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?," *Proc. Computer Human Interaction 2000,* pp. 345-352, 2000.

[35] C. Ware, "Using Hand Position for Virtual Object Placement," *Visual Computer,* vol. 6, pp. 245-253, 1990.

[36] I. Poupyrev, S. Weghoast, and S. Fels, "Non-Isomorphic 3D Rotational Techniques," *Proc. Computer Human Interaction 2000,* pp. 540-547, 2000.

Knud Henriksen received the BS degree in electrical engineering from Odense Teknikum in 1975, the BS degree in mathematics and computer science from the University of Copenhagen in 1979, the MS degree in computer science in 1986, and the PhD degree in computer science in 1990, from the same university. He is an associate professor in the Department of Computer Science, University of Copenhagen. From 1984-1986, he was employed as a research assistant at the GRASP lab, University of Pennsylvania, Philadelphia. His interests include interactive computer graphics, animation, and dynamic simulation.

Jon Sporring received the master's and PhD degrees from the Department of Computer Science, University of Copenhagen, Denmark, in 1995 and 1998, respectively. Part of his PhD program was carried out at the IBM Research Center, Almaden, California. Following receipt of his PhD, he worked as a visiting researcher at the Computer Vision and Robotics Lab at the Foundation for Research & Technology-Hellas, Greece, and as assistant research professor at the 3D-Lab, School of Dentistry, University of Copenhagen. Since 2003, he has been employed as an associate professor in the Department of Computer Science, University of Copenhagen. His main topic of research is medical image processing, medical computer graphics, and information theory.



Kasper Hornbæk received the MSc and PhD degrees in computer science from the University of Copenhagen in 1998 and 2001, respectively. He is an assistant professor at the Natural Sciences ICT Competence Center at the University of Copenhagen. His research interests include human-computer interaction, information visualization, e-learning, and computer-supported collaborative learning.