Extending the Virtual Trackball Metaphor to Rear Touch Input

Sven Kratz[*]
Deutsche Telekom Laboratories, TU Berlin

Michael Rohs[†]
Deutsche Telekom Laboratories, TU Berlin

ABSTRACT

Interaction with 3D objects and scenes is becoming increasingly important on mobile devices. We explore 3D object rotation as a fundamental interaction task. We propose an extension of the virtual trackball metaphor, which is typically restricted to a half sphere and single-sided interaction, to actually use a full sphere. The extension is enabled by a hardware setup called the "iPhone Sandwich," which allows for simultaneous front-and-back touch input. This setup makes the rear part of the virtual trackball accessible for direct interaction and thus achieves the realization of the virtual trackball metaphor to its full extent. We conducted a user study that shows that a back-of-device virtual trackball is as effective as a front-of-device virtual trackball and that both outperform an implementation of tilt-based input.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies, interaction styles, haptic I/O

1 INTRODUCTION

Interaction with 3D objects and scenes has many applications on mobile devices, including games and 3D model inspection. The hardware capabilities of mobile devices for rendering 3D content are increasing rapidly. Direct touch input on mobile devices potentially allows for more direct interaction with 3D objects than is possible on desktop PCs. However, there are unique challenges of devices with small displays, such as the relatively large amount of occlusion that occurs with direct finger-based touch interaction. This problem has been named the "fat finger problem" [1]. Several solutions have been proposed for overcoming or alleviating this issue, including back-of-device interaction [1, 11, 13, 14], tilting-based interfaces [10], and temporal separation of input event and interface action [4].

Relatively little research has focused on mobile interaction with 3D content (an example is [4]). In this work we are focusing on 3D rotation, which is a fundamental task allowing users to inspect and understand 3D objects. A widely used interaction technique for rotating 3D objects around arbitrary axes is the virtual sphere [3], also called the virtual trackball [6]. A further related technique to be mentioned is the Arcball [12]. In this technique, rotation is controlled by projecting cursor points from the display surface onto a virtual half sphere that is centered on the 3D object (Figure 1). Rotation around the object's center is then computed using the projected points. The virtual trackball is typically invisible and does not have to be known to the user in order to be applied effectively. Virtual trackballs allow for rotation around arbitrary axes in a natural manner. They integrate controller and controlled object and can thus be categorized as a direct manipulation technique [7]. In user tests virtual trackballs have been shown to be effective for precise 3D rotation tasks [3]. Other work ([9] and references therein) investigates multi-finger 3D interaction for large screens.

[*]e-mail: sven.kratz@telekom.de
[†]e-mail:michael.rohs@telekom.de

We propose to extend the virtual trackball metaphor, which is typically restricted to a half sphere and single-sided interaction, to actually use a full sphere. In order to achieve this we use an implementation called the "iPhone Sandwich," which allows for simultaneous front-and-back touch input (Figure 6). This setup makes the rear part of the trackball accessible for direct interaction and allows for a natural implementation of the virtual trackball metaphor to its full extent.

This paper is structured as follows. In the next section we discuss the details of the two-sided virtual trackball design. We then discuss our particular implementation and present a user study on 3D object inspection that compares front and back virtual trackballs and tilt-based control.

2 INTERACTION CONCEPT

We implemented the virtual trackball metaphor for direct touch input on mobile devices and extended it for back-of-device input. Traditionally the mapping is done in such a way that a touch point $p_a = (x_a, y_a)^\top$ is mapped onto a 3D point $m(p_a) = P_a = (x_a, y_a, z_a)^\top$ on a hemisphere (if the distance from p_a to the origin o is less than the radius r of the hemisphere) using an orthographic projection $m : \mathbb{R}^2 \to \mathbb{R}^3$. If the touch point is outside of the hemisphere, then it is mapped to $P_a = (x_a, y_a, 0)^\top$. (The image plane is given by $z = 0$.)

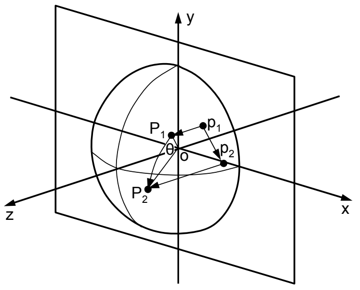

Figure 1: Touch points (p_1, p_2) inside virtual hemisphere: rotation defined by great circular arc through projected touch points (P_1, P_2).

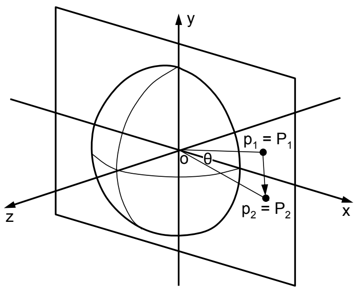

Figure 2: Touch points (p_1, p_2) outside virtual hemisphere: rotation around z-axis.

The axis and angle of rotation are determined as follows. Assume that p_2 is the current touch point and p_1 is the previous touch point in a sequence of touch points produced during a stroke. The axis of rotation is then the cross product of the projected points: $a = m(p_1) \times m(p_2) = P_1 \times P_2$ and the angle of rotation is $\theta = \arccos(\frac{P_1 P_2}{\|P_1\|\|P_2\|})$. This means that the rotation will be around the z-axis if p_1 and p_2 are both outside the circle (Figure 2). If both projected points are located on the hemisphere, then the rotation axis is given by the great circle passing through P_1 and P_2 (Figure 1).

2.1 Two-Sided Trackball and Gaussian Mapping

As noted in [6] there is a discontinuity between positions inside and outside the hemisphere. In our pilot tests we found that the change in rotation behavior on entering or leaving the hemisphere during a drag movement can indeed be irritating to users. To avoid this discontinuity and to have a smoother transition from z-axis rotation to x- or y-axis rotation, we changed the mapping from a hemisphere to a Gaussian function (Figure 4, upper part), similar to the combined spherical and hyperbolic mapping function of Bell [6]. The specific mapping function we use is: $z = g(x,y) = r_1 \exp(\frac{-((x-o_x)^2+(y-o_y)^2)}{2r_2^2})$, where $o = (o_x, o_y)$ is the center of the screen. Note that we do not need to distinguish between the cases of the point being located inside the hemisphere or outside of it. Exclusive rotation in the z axis thus results if input is performed at a sufficient distance from the center of the virtual trackball. The parameter r_1 corresponds to the height of the Gaussian curve and r_2 to its width. For the 480×320 pixel screen of our target device we set these parameters to $r_1 = 120$ and $r_2 = 80$, respectively. A precise correspondence to the size of the object was not found to be necessary.

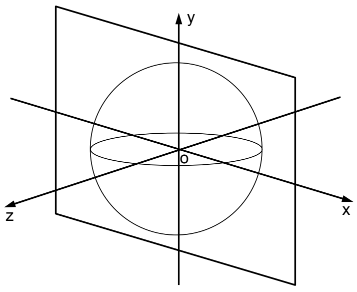

Figure 3: Extension of half sphere to full sphere enables rotation control from both front and back.

To allow for back-of-device interaction we conceptually extended the half sphere to the backside. For the back side mapping function we inverted the above Gaussian function to extend its range into negative z values as shown in Figure 4 (lower part). Depending on whether the touch input event originates from the front side or the back side, the corresponding mapping function is used. For front-side touch events the mapping function is $z = g(x,y)$, for back-side touch events it is $z = -g(x,y)$.

In addition to the single sided touch input described above and used in the study, we also implemented a simultaneous front- and back-touch input. In this mode the thumb provides the front touch point and the index finger provides the rear touch point. The center of the virtual sphere is then relocated to be in the center between thumb and index finger. This relocation of the virtual sphere allows for 3D rotation in the case of larger displays in which the center of the object might not be reachable with simultaneous front- and back-input.

2.2 Tilt-Based Navigation

In addition to touch screen input we implemented a tilt-based scheme that uses an accelerometer to sense device orientation (Figure 5). The motivation for choosing tilt-input is the increasing popularity of tilt-based gaming applications on certain modern smartphones. The navigation is based on a steering-wheel metaphor, in which the object rotates horizontally as the user tilts the device around the z-axis. Horizontal rotation of the object is initiated if the device tilt exceeds $\pm 25°$ around the z-axis. The rotation speed was set to be proportional to the tilt angle of the z-axis, with a minimum rotation speed of $100°/s$ at $25°$ tilt and a maximum rotation speed of $350°/s$ at $90°$ tilt. Pre-tests showed that users could comfortably operate the tilt interface with these settings. Moreover, the amount of vertical camera rotation is controlled by tilting around the x-axis. This determines the angle from which the object is observed. The virtual object is always oriented with the right side up, i.e. the y-axis is always oriented opposite to the direction of gravity.

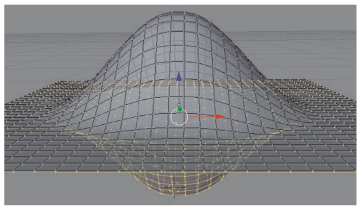

Figure 4: Gaussian function instead of half sphere to avoid discontinuity. Extends virtual trackball metaphor for front and back touch input.

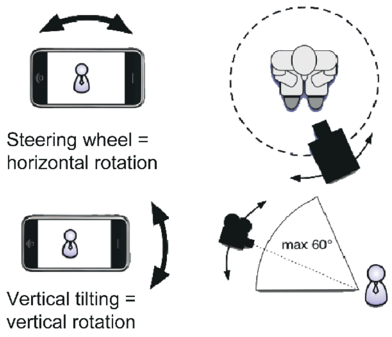

Figure 5: Tilt-based navigation using tilt around the z-axis (horizontal camera rotation) and x-axis (vertical camera rotation).

3 IMPLEMENTATION

The front and rear touch interface for our study was implemented on an iPhone "Sandwich" (Figure 6), which consists of two Apple iPhones (2G version) linked back-to-back in order to enable back of device interaction. The software consists of a dedicated *rear* application that runs that sends rear touch events to the front device, which runs the *front* application used in the user study. The data is transferred via UDP over a dedicated WiFi network. There was no perceivable delay. The front and rear virtual trackballs were implemented using the touch events provided by the front and rear iPhone, respectively.

Device tilt readings for x ($\mathrm{tilt_x} = \arctan\left(\frac{z}{x}\right)$) and z ($\mathrm{tilt_z} = \arctan\left(\frac{y}{x}\right)$ axes obtained from acceleration data provided by the iPhone's built-in 3 axis acceleration sensor. z represents the acceleration measured perpendicular to the device's screen, x the acceleration in direction of the long side of the device and y the acceleration in direction of the short side of the device.

The graphics for the *front* application were implemented using OpenGL ES [8]. We used Blender [2] to model and export the 3D objects with corresponding normal vectors and texture mapping

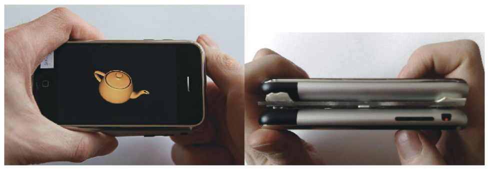

Figure 6: The iPhone "Sandwich" as used in the study.

coordinates. Via a UDP connection, the experimenter can change the number of objects displayed as well as the number of textured faces shown on the objects in the scene. Through this connection, trials can also be started and stopped remotely.

4 User Study

We conducted a user study in order to measure the effectiveness of 3D object rotation using a front and rear touch virtual trackball as well as tilt. We extended the experimental setup of Decle et al. [4], who compared direct and planned 3D object inspection on a touch display.

4.1 Task

Following the approach proposed by Decle, the test subjects were presented with a freely rotatable 3D scene comprising a regular grid of tetrahedrons. The subjects had to count the number of object faces textured with a logo. Each face of the tetrahedrons was colored in a distinct color, which allowed the test subjects to remember the sides of the objects in a given scene which had already been observed.

Initially, a blank screen was presented to the test subjects. The experimenter could remotely change the settings for grid size and number of textured faces remotely as well as initiate a trial. Once the trial had been initiated, the corresponding scene was presented to the test subject and a timer started. When the test subjects were satisfied that they had found all the textured faces, they reported the number of textured faces found to the experimenter. The task completion time and number of found textured faces found was recorded by the experimenter after each trial.

This type of task is well suited to evaluate 3D rotation input techniques, because it requires the test subjects to look at all the faces of the objects, thus requiring a substantial amount of rotation input from the subjects. The effectiveness of the rotation technique can be deduced from the trial completion times (input speed) and also the error rates (an indicator of mental load).

4.2 Improvement to Existing Methodology

In contrast to the previous work, which used a randomly chosen object in their study, we used a regular grid of tetrahedrons (Figure 7) as object set. The number of objects and the amount of textured faces that must be found by the test subjects can be programmatically defined. This not only allows us to precisely parametrize the characteristics of the experiment, it also provides a more controllable and comparable scenario for comparison of rotation techniques in future studies.

4.3 Participants, Apparatus, and Design

We recruited 12 test subjects from a university environment. All participants were between 20 and 25 years of age. They all owned a mobile phone, but only one test subject reported to own a touch-enabled smart phone. Only two of the subjects had prior experience with mobile 3D applications. All participants had experience in the use of computers, with 42 percent of them stating advanced or expert skills. The participants received a monetary compensation after the experiment.

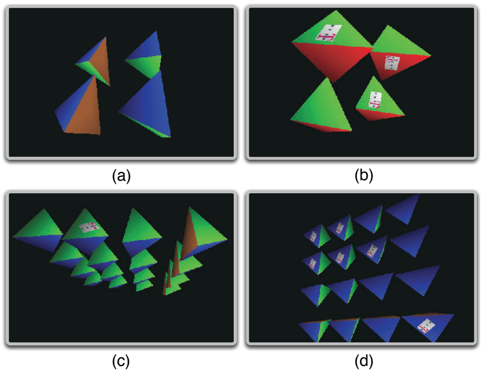

Figure 7: Several variants of the tetrahedron grid presented to the user in the study are shown. (a) is 2×2 grid with no textures. (b) is 2×2 grid with 3 of 5 textures visible to the user. (c) shows a 4×4 grid with a single visible texture. (d) shows a 4×4 grid with 6 of 10 textures visible.

The experiment was conducted using the iPhone Sandwich discussed in the previous section. For the trials involving the front and rear trackball and a standard iPhone 3G for the trials involving the tilt input.

The experiment had a 3×2×2 within groups factorial design. Factors were *input method* for rotation control (front trackball, rear trackball, and tilt), *grid size* (2×2 and 4×4) as well as textured *face count* (3 ± 1 and 9 ± 1). The textured face count was randomly chosen in a ± 1 range around 3 and 9 in order to prevent the test subjects from inferring the correct number of textured faces and forcing them to really count all textured faces in the scene presented to them. The order of input techniques was counterbalanced according to a Latin Square design. The trials for each input technique were conducted in sequence, with the order of the grid size and textured face count settings also counterbalanced according to a Latin Square. Each setting was repeated two times resulting in a total of $12 \times 3 \times 2 \times 2 \times 2 = 288$ trials conducted.

We measured the task completion time as well as the error rate of the reported textured face counts. Additionally, after each series of trials for a given input technique, the participants were asked to subjectively rate the workload of the input method using the NASA TLX [5] rating scale.

4.4 Results

4.4.1 Task Completion Times

Figure 8 shows box plots of the task completion times grouped by input method, grid size and number of textured faces.The mean execution time for *front* was $14.72s$, SD $= 7.51s$, for *rear* $13.89s$, SD $= 6.56$ and for *tilt* $23.73s$, SD $= 12.60s$.

A univariate ANOVA shows significant effect in the task execution time for input method ($F_{2,287} = 38.73, p < 0.001$), grid size ($F_{1,287} = 34.722, p < 0.001$) and number of textured faces ($F_{1,287} = 16.820, p < 0.001$). A Bonferroni pairwise comparison of input method shows a significant difference for *front* vs. *tilt* and *rear* vs. *tilt* (both $p < 0.001$), but no significant difference between *front* and *rear* ($p = 1.0$).

The error rate, i.e. the number of incorrect responses to the total number of responses, was 60.4% for *front*, 69.8% for *rear*, and 68.6% for *tilt*. Participants were not provided with feedback whether they counted the right number of textured surfaces, because we wanted to measure neutral error performance. The error rates appear quite high, however, the responses were very close to the actual numbers. The mean square error, i.e. the deviation of the reported count from the actual count, was 0.97 for *front*, 0.70 for

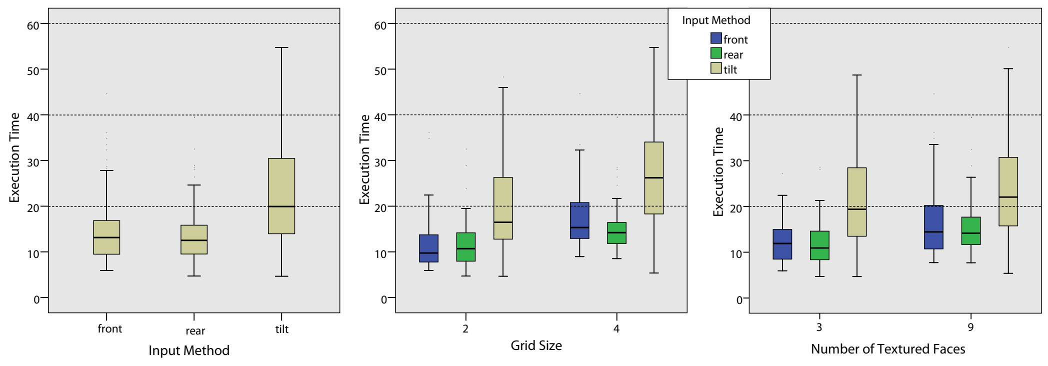

Figure 8: Box plots of the task completion times in s, grouped by input method (*left*), grid size (*center*) and number of textured faces (*right*).

rear, and 0.94 for *tilt*.

4.4.2 NASA TLX

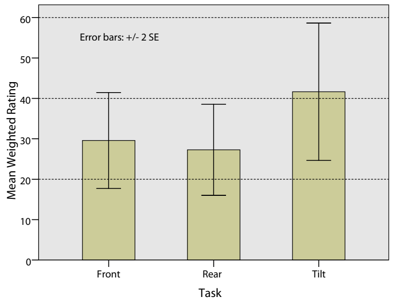

Figure 9: The average adjusted workload of the TLX rating scale.

The average adjusted NASA TLX workload ratings are shown in Figure 9. *Front* and *Rear* received very similar workload ratings (*front*: AVG=29.6, SD=20.6; *rear*: AVG=27.3, SD=19.5). *Tilt* was rated worse (AVG=41.6 SD=29.4). These differences are however only indicative and not statistically significant ($F_{2,35} = 1.28, p = 0.29$).

5 Conclusion

In this paper we propose the extension of the virtual trackball metaphor from single-sided input on a half-sphere to double-sided input on a full sphere. We show an alternative Gaussian mapping function that avoids the discontinuity of a spherical trackball, similar to Bell's [6] spherical and hyperbolic mapping function. In a user study we found that the virtual trackball with this mapping function is effective for front as well as back touch input and that both outperform a tilt-controlled navigation scheme. Even though the tilt condition avoids occlusion, it is rate controlled and has a more indirect mapping compared to the virtual trackball. NASA TLX showed a slightly higher workload for tilt, however, without being statistically significant. Overall, it is noteworthy that the subjective workload rating was quite low (the scale ranges from 0 to 100, the maximum rating was 41.6 for tilt) yet the error rate was relatively high. This means that the participants had difficulty in judging their true performa nce and the result might be an indication that 3D object inspection is inherently difficult. Our results also indicate that occlusion, or "fat finger problem" did not have a significant performance effect on the tasks we measured. This may have been due to the relative rotation of the 3D objects realized with the virtual trackball, which allowed the users fingers to operate at an offset with respect to the rotated objects. In the future we plan to extend this work by adding precision alignment studies. We also look to investigate the use of multiple fingers on one side of the device as well as the use of fingers on both sides of the trackball. Furthermore we look to integrate the behind-the-back trackball paradigm in mobile applications scenarios.

References

[1] P. Baudisch and G. Chu. Back-of-device interaction allows creating very small touch devices. In *Proc. of CHI '09*, pages 1923–1932, New York, NY, USA, 2009. ACM.

[2] Blender.org. Blender. `http://www.blender.org/`.

[3] M. Chen, S. J. Mountford, and A. Sellen. A study in interactive 3-d rotation using 2-d control devices. In *Proc. of SIGGRAPH '88*, pages 121–129, New York, NY, USA, 1988. ACM.

[4] F. Decle and M. Hachet. A study of direct versus planned 3d camera manipulation on touch-based mobile phones. In *Proc. of MobileHCI '09*, pages 1–5, New York, NY, USA, 2009. ACM.

[5] S. Hart and L. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human mental workload*, 1:139–183, 1988.

[6] K. Henriksen, J. Sporring, and K. Hornbaek. Virtual trackballs revisited. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):206–216, 2004.

[7] E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct manipulation interfaces. *Hum.-Comput. Interact.*, 1(4):311–338, 1985.

[8] Khronos Group. OpenGL ES. `http://www.khronos.org/opengles/`.

[9] J. L. Reisman, P. L. Davidson, and J. Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78, New York, NY, USA, 2009. ACM.

[10] J. Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 167–168. ACM Press, 1996.

[11] E.-l. E. Shen, S.-s. D. Tsai, H.-h. Chu, Y.-j. J. Hsu, and C.-w. E. Chen. Double-side multi-touch input for mobile devices. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4339–4344, New York, NY, USA, 2009. ACM.

[12] K. Shoemake. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, volume 92, pages 151–156, 1992.

[13] M. Sugimoto and K. Hiroki. Hybridtouch: an intuitive manipulation technique for pdas using their front and rear surfaces. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 137–140, New York, NY, USA, 2006. ACM.

[14] D. Wigdor, C. Forlines, P. Baudisch, J. Barnwell, and C. Shen. Lucid touch: a see-through mobile device. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 269–278, New York, NY, USA, 2007. ACM.