An Investigation into Round Touchscreen Wristwatch Interaction

Daniel Ashbrook, Kent Lyons[†], Thad Starner

College of Computing and GVU Center
Georgia Institute of Technology
Atlanta, GA 30332
{anjiro,thad}@cc.gatech.edu

[†]Intel Research
2200 Mission College Blvd.
Santa Clara, CA 95054
kent.lyons@intel.com

ABSTRACT

The wristwatch is a device that is quick to access, but is currently under-utilized as a platform for interaction. We investigate interaction on a circular touchscreen wristwatch, empirically determining the error rate for variously-sized buttons placed around the rim. We consider three types of inter-target movements, and derive a mathematical model for error rate given a movement type and angular and radial button widths.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Theory and methods

General Terms

Human Factors

1. INTRODUCTION

Research has shown that the amount of time it takes to access a device has a strong influence on whether a user will actually use that device at all [3, 7]. Previous research [1] has demonstrated that the wrist is an excellent location to place devices that need to be accessed quickly. The question that is thus far left unanswered is how to create a usable interface on the wrist. In this paper, we consider touchscreen watch interfaces.

Touch is an intuitive interaction method, and in the past decade touchscreens have become very popular for mobile devices. The number of commercially-released touchscreen wristwatches, however, has been very small, and they have in general not been successful. We believe that there are several reasons for this, two of which we address in this research.

The first issue with extant touchscreen watches is button size. All commercially released touchscreen watches thus far have shipped with a tiny stylus to allow the user to manipulate the onscreen interface, which invariably has minuscule buttons. Users, however, prefer to use their fingers to manipulate touchscreens, to avoid the time needed to retrieve the stylus [8].

A second issue with current touchscreen watches is lack of style. Wristwatches are often worn for fashion, but technological wristwatches appear to seldom fulfill this need. One reason may be that round wristwatches are preferred over rectangular ones. Indeed,

many digital watches still have round faces surrounding the blocky digital time readout.

To address the issues of finger-usability and style inherent in current touchscreen wristwatches, we are investigating potential interfaces for touchscreen wristwatches with circular displays. We believe that a round-faced watch will be more accepted by consumers, and by designing the interface to be used by fingers—rather than by a stylus—we can make the watch easier to use.

In this paper, we will present our preliminary findings on interaction with a round touchscreen watch. Inspired by non-wristwatch circular interfaces such as Pie Menus [2], we consider the effects of placing wedge-shaped buttons around the perimeter of a circular screen. Most circular interfaces in the literature have been implemented for screens PDA-sized or larger, where the size of the buttons is less of a concern. On a wristwatch-sized screen, however, there will be a tradeoff between the number of buttons that can fit around the edge and the usability of the device; additionally, there may be a tradeoff between usability and the amount of non-button area left in the center for display purposes. Figure 1 shows an example of the kind of application we hope to enable with this research.
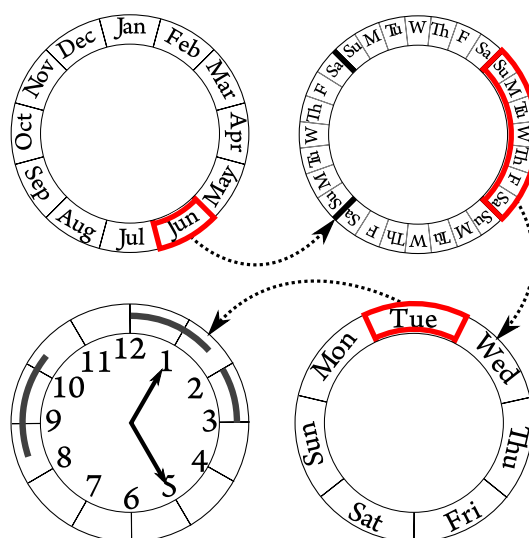

Figure 1: Mockup of a potential zooming interface for a calendar, using the *rim* style of interaction. The user selects the area denoted with the thick red line and "zooms" to the next level of detail with a sweep of the finger around the bezel. Each circle is printed at actual size.
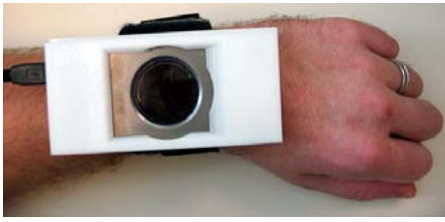
Figure 2: The simulated touchscreen watch.

Like rectangular screens, circular screens will necessarily have a bezel in order to accommodate the screen's electronics. As with the work by Froehlich *et al.* [4] and the work by Blaskó [5] we investigate the possibility of using the bezel to help guide the user's finger. We chose to study three types of finger/screen interactions that might be used on a round touchscreen watch: 1) tapping, as with a standard PDA-like application; 2) sliding in a straight line; or 3) sliding along the rim of the watch, using the bezel as a guide, as in [4] and [5].

2. EXPERIMENT

We conducted an experiment in order to investigate the size of buttons necessary to allow finger-based interaction on a small, round touchscreen watch. In particular, we wanted to understand the tradeoffs between number of buttons, non-button area remaining in the center, and error rate for each of the the three types of movement we discussed above.

2.1 Participants

Fifteen volunteers were recruited to participate in the experiment. Fourteen volunteers were male and one was female; ages ranged from 20 to 28. Participants were paid US $10 per hour for participation; no participant took more than one hour to complete the experiment. All participants were either right-handed or primarily used their right hand for control tasks such as mouse movement. Eight participants reported normally wearing a watch, and each stated they normally wore it on their left wrist.

2.2 Variables

Movement type is our primary independent variable. The simplest method is using a tap to select, much like using a mouse to click on icons. This is the *tap* condition. Another method we investigated is sliding the finger in a straight line from one place on the surface to another; this forms the *through* condition. The final option is sliding the finger using the bezel as a guide, avoiding the center region of the watch. This is the *rim* condition.

The main dependent variables of interest are *button size* and *error rate*. Each button has both an *angular* and a *radial* width. As the buttons increase in angular width, fewer buttons will fit around the rim of the watch. As the radial width increases, less space will be available in the center of the watch for a display or other non-button purposes.

2.3 Apparatus

Lacking a programmable round touchscreen watch, we simulated one. We removed the case from a Motorola E680i touchscreen mobile phone and placed it into a custom-made plastic case (Figure 2). In order to simulate a round watch face with a bezel, we placed over the screen a custom-cut plate of 1.6mm (.06in) thick steel with a hole approximately 30mm (1.25in) in diameter, with a sloping bevel around the hole. The hole revealed an area of the phone's screen 200 pixels in diameter. The "watch" was mounted

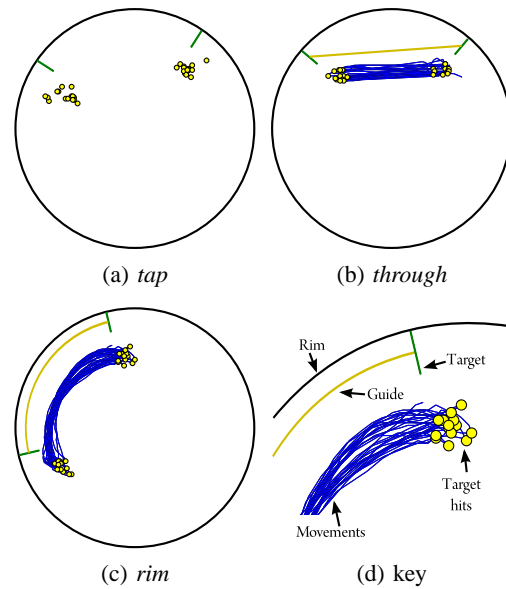(a) *tap* (b) *through*

(c) *rim* (d) key

Figure 3: Targets with completed conditions for one participant. Rim, target and guide were displayed to participant as shown in (d); movements and target hits were displayed only to researcher. Images (a)–(c) are printed at actual size.

on each participant's arm using a velcro strap, with the visible portion of the screen centered on the arm approximately where a normal wristwatch face would be. We wrote software for the phone to present trials to participants and collect data.

2.4 Procedure

Our experiment is a 3×12 within-subjects factorial design. Participants performed a Fitts-style reciprocal 2D pointing task using their finger as the pointer. Two targets were displayed on the face of the wristwatch, and a secondary monitor instructed the participants to select the targets as quickly and accurately as possible, using the *tap*, *through* or *rim* movement type. For the *through* and *rim* conditions, a line was drawn on the watch face as a reminder of the motion to make (Figure 3). No feedback was given to participants during the trials, except to blank the screen to indicate the end of each trial.

2.5 Design

Each volunteer participated in 108 trials. The primary independent variables were movement type (*tap*, *through* and *rim*) and distance between targets (twelve distances were used, as explained below). To avoid learning effects, conditions were presented to participants in a balanced latin square ordering.

In order to avoid biasing participants with any particular size of target, and following the literature [6], we selected targets as close to zero width as we could display. These were simple lines extending from the outside of the watch 18 pixels (2.7mm, .11in) towards the center (Figure 3). Each line was one pixel (.15mm) wide, or approximately $0.58°$ at the inner end of the target. While these targets are unrealistically small, they will allow us to consider the distribution of nearby hits.

To determine target positions, we divided the face of the watch into twelve (hour-sized) segments of $30°$ each, and then further subdivided each segment in half to get a minimum distance between targets of $15°$. At the inner end of the targets, this is a about 21 pixels, or 3.2mm (.13in). We tested participants on all of the $15°$
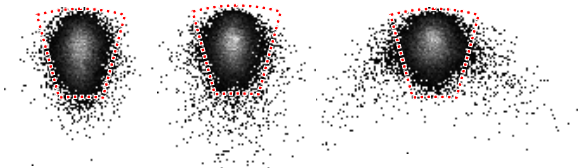
Figure 4: Distributions achieved by rotating clusters to 0°. Shown from left to right are points from *tap*, *through*, and *rim* conditions. Lighter colors denote a higher density of endpoints. Red dotted lines indicate example simulated button of 30° angular width and 50 pixels radial width.

increments between 15° and 180°, inclusive, for a total of twelve different distances. After choosing the distance between each pair of targets, the pair was centered at the rim of the watch on a location chosen from a uniform random distribution from 0–360°. This ensured that the data would cover the face of the watch.

Of note is that while each movement condition used the same set of target spacings, these distances differ between the *rim* and the other two conditions: a 180° distance in the *tap* or *through* condition will yield a 200 pixel (30mm) straight-line distance, but in the rim condition the participant's finger must travel in an arc, yielding a movement equal to πr, or 314 pixels (47.1mm). We chose to maintain equivalent target spacings between conditions to mimic the effects of an interface where relative button distances would be the same regardless of the method used to activate the buttons.

During each trial, the participant moved back and forth between the pair of targets 15 times, resulting in 30 movement end points. The end points in the *through* and *rim* conditions were determined by the user reversing movement direction; in the *tap* condition any tap on the screen was considered to be an end point. The software counted the end points and automatically ended the trial when the goal was reached. A block is a set of twelve trials (one per distance) with one movement type (*e.g. rim*); the blocks were repeated three times for each of the three movement types. Therefore we have 12 distances × 3 repetitions × 3 movement types = 108 trials per participant. In between each block, participants were given an enforced 30-second break to avoid fatigue; if desired, they could break for longer than 30 seconds.

2.6 Results

After the experiment was complete, we processed and analyzed the data. Because we did not force participants to actually hit each target before proceeding to its opposite, it was sometimes difficult to determine which target the participant had been trying to hit, especially at small inter-target distances. Therefore, we used the k-means clustering algorithm to separate the points into two groups. As the seed means for the clusters, we used the innermost tips of the target lines. After clustering, the standard deviation (σ) of each cluster was calculated, and points falling greater than 3σ away from the cluster mean were discarded.

We next took the clusters from all of the participants and grouped them by movement type. We then rotated all of the clusters from each movement type so that the mean of each cluster was at 0°, giving an overall distribution for that movement type. The results are illustrated in Figure 4. We then calculated error rates for different radial and angular button widths. We accomplished this by creating simulated buttons (red dotted lines in Figure 4) for every integer combination of 0–100 pixels of radial width and 0–100° of angular width. Each button was placed against the rim and centered on 0°. The error rate for that button was then calculated simply by counting the number of inflection points falling inside the button versus
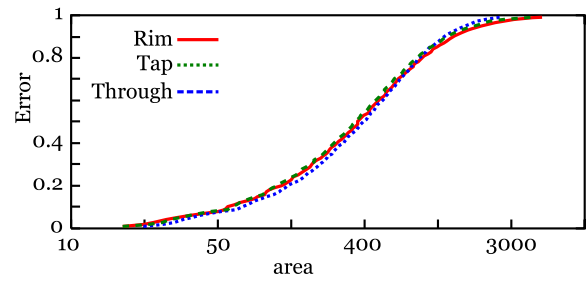

Figure 5: Button area (Equation 1) vs error rate. Note that *x*-axis is a log scale.
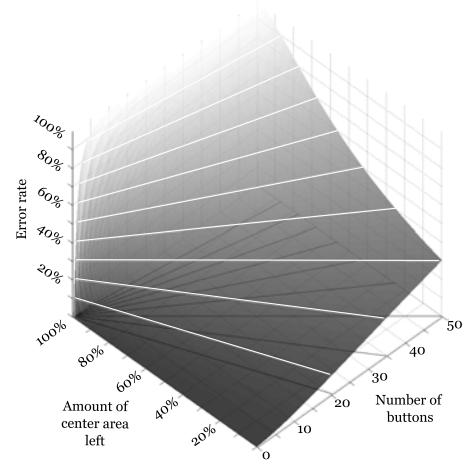

Figure 6: Number of buttons to be placed around the edge *vs* amount of surface area to be left for center area *vs* calculated error rate (Equation 2) using "all" row of Table 1. Error rate contour lines are displayed on the surface at 10% increments (solid white lines) and projected downwards (solid black lines).

those falling outside. For each combination of radial and angular widths, we also calculated the area (in pixels) of the button:

$$area = \frac{ang}{360}\pi\left(R^2 - (R-rad)^2\right) \tag{1}$$

where *R* is the radius of the watch (100 pixels), *ang* is the angular width of the button, and *rad* is the radial width. This area is equivalently expressible in terms of the number of buttons placed around the edge *numbt* and the percentage of center area (relative to the entire surface area) available for other purposes *pctctr*:

$$area = \frac{\pi R^2}{numbt}\left(1 - pctctr\right)$$

By plotting the error rate (0–100%) against the log of the area of the button that resulted in that error rate, we observe a sigmoidal shape (Figure 5). We performed a least-squares fit to the data using the equation

$$1/\left(1 + x^{y - \ln(area)}\right) \tag{2}$$

where *area* is the button area as calculated in Equation 1. Close fits were found, as reported in Table 1.

The close fit of the data to Equation 2 allows us to model error rates for given angular and radial widths. A visualization of predicted error rates for one to fifty buttons with sizes such that zero to one-hundred percent of the center area remains is displayed in Figure 6, with contour lines at 10% error increments.

Because of finger width, participants could not touch the screen at the full radius; for the *tap* condition, the average distance away

Condition	x	y	R^2	RMSE
tap	4.895	5.932	0.9939	0.0225
through	4.382	5.831	0.9964	0.0172
rim	4.247	5.878	0.9970	0.0157
all	4.476	5.881	0.9939	0.0224

Table 1: Fits of error rate *vs* log of area data to Equation 2. The "all" condition represents the data for all three conditions aggregated together.

from the rim was 23.8 pixels ($SD = 6.8$), for *through* 21.1 pixels ($SD = 7.2$), and for *rim* 17.7 pixels ($SD = 6.7$). We measured the amount of time for participants to move between targets; perhaps because of the small distances involved, there was no correlation between inter-target movement time and distance. Mean per-condition movement times are *tap*: 0.32s ($SD = 0.08$); *through*: 0.27s ($SD = 0.14$); *rim*: 0.39s ($SD = 0.19$).

2.7 Discussion

Given the equations above, we can now think about how a wristwatch interface might look. Because participants were requested to move between the targets as quickly and accurately as possible, the error rates can be assumed to be worst-case. By taking into account the desired application or situation, the correct number of buttons and center area can be found for a desired error rate. For example, if we want an error rate of 5% for ten buttons, we will have about 90% of the surface area left for the display. Figure 7 illustrates several possibilities for different combinations of button size and error rate based on our results.

As a general rule, it was difficult for participants to get their fingers close to the rim; this is reflected in the sharp slope nearing 100% in the "center area" axis of Figure 6. We would therefore recommend keeping the radial width of buttons above the mean distance from the edge for each movement type.

3. CONCLUSIONS & FUTURE WORK

The results of our experiment yielded a mathematical model of error rate given the angular and radial widths for buttons placed around the edge of a circular touchscreen watch. We determined constants x and y for the model for three inter-target movement types: tapping (*tap*), sliding in a straight line between targets (*through*) and sliding along the bezel of the display (*rim*).

Our experiment gave us a wealth of data and many directions for future exploration. We wish to experimentally validate our results by running another study with various button sizes to ascertain how well our model predicts error rate with real buttons, rather than simple lines.

We intend to investigate whether there is a bias in error rate based on the location of the target on the watch: observation during the experiment suggested that targets in the upper-left quadrant of the watch—from 9 o'clock to 12 o'clock—are more likely to be obscured by the user's finger, while targets on the bottom rim—from 4 o'clock to 8 o'clock—may be more difficult to hit due to the shape of the user's finger.

Other work has addressed the issue of screens obscured by fingers, for example the Shift offset cursor technique by Vogel and Baudisch [8]; however, such techniques usually rely on having extra screen real-estate where the obscured content can be displayed. We are considering methods for accomplishing a similar interaction style in our extremely limited display space, such as mirroring the content across the face of the watch or using fisheye views.

The ultimate motivation for this work is to allow fast access and use of the watch in any situation, including while actually in mo-
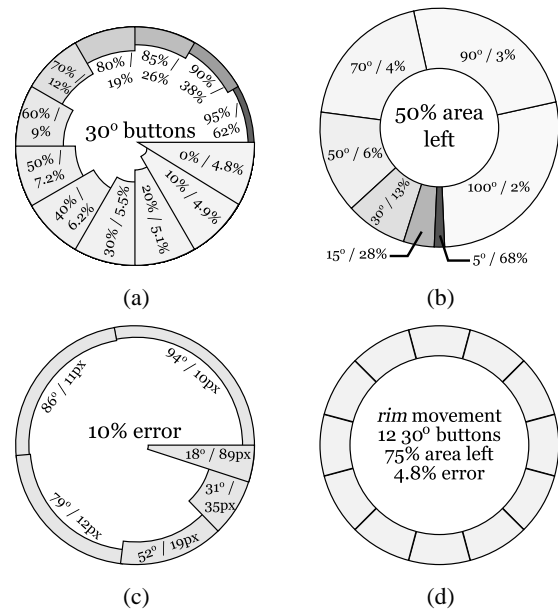

(a)   (b)

(c)   (d)

Figure 7: (a)–(c) illustrate holding constant angular width (a), radial width (b) or error rate (c), using numbers from the "all" row in Table 1. For (a) and (b) the first number in each wedge represents percent of center area left and radial width, respectively, and the second number is the error rate; while for (c) the two numbers are angular (degrees) and radial (pixels) width. (d) illustrates a possible layout using the *rim* numbers from Table 1. Each circle is printed at the actual size of the watch face used in the experiment.

tion. We are particularly interested in whether the stabilization offered by the bezel in the *rim* movement condition confers an advantage over the other two movement types while the user is in motion.

Finally, we plan on implementing some interfaces, such as illustrated in Figure 1, using our results and performing user studies to determine their efficacy.

Acknowledgements

4. REFERENCES

[1] D. Ashbrook, J. Clawson, K. Lyons, N. Patel, and T. Starner. Quickdraw: The impact of mobility and on-body placement on device access time. In *CHI*, 2008.

[2] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *CHI*, 1988.

[3] Y. Cui, J. Chipchase, and F. Ichikawa. A cross culture study on phone carrying and physical personalization. In *HCI International*, 2007.

[4] J. Froehlich, J. Wobbrock, and S. Kane. Barrier pointing: using physical edges to assist target acquisition on mobile device touch screens. In *ASSETS*, 2007.

[5] Gábor Blaskó. *Cursorless Interaction Techniques for Wearable and Mobile Computing*. PhD thesis, Columbia University, 2007.

[6] R. Schmidt, H. Zelaznik, B. Hawkins, J. Frank, and J. Quinn. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86(5), 1979.

[7] T. Starner, C. Snoeck, B. Wong, and R. McGuire. Use of mobile appointment scheduling devices. In *CHI*, 2004.

[8] D. Vogel and P. Baudisch. Shift: A technique for operating pen-based interfaces using touch. In *CHI*, 2007.