Semi-Automatic Zooming for Mobile Map Navigation

Sven Kratz[1], Ivo Brodien[2], Michael Rohs[1]
Deutsche Telekom Laboratories, TU Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
[1]{sven.kratz, michael.rohs}@telekom.de
[2] philotas@cs.tu-berlin.de

ABSTRACT

In this paper we present a novel interface for mobile map navigation based on Semi-Automatic Zooming (SAZ). SAZ gives the user the ability to manually control the zoom level of an SDAZ interface, while retaining the automatic zooming characteristics of that interface at times when the user is not explicitly controlling the zoom level. In a user study conducted using a realistic mobile map with a wide scale space, we compare SAZ with existing map interface techniques, multi-touch and Speed-Dependent Automatic Zooming (SDAZ). We extend a dynamic state-space model for Speed-Dependent Automatic Zooming (SDAZ) to accept 2D tilt input for scroll rate and zoom level control and implement a dynamically zoomable map view with access to high-resolution map material for use in our study. The study reveals that SAZ performs significantly better than SDAZ and that SAZ is comparable in performance and usability to a standard multi-touch map interface. Furthermore, the study shows that SAZ could serve as an alternative to multi-touch as input technique for mobile map interfaces.

Keywords

mobile devices, tilt input, map navigation, zooming-scrolling UI, automatic zoom, dynamics, SDAZ

General Terms

Algorithms, Experimentation, Performance, Measurement

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: Input devices and strategies.

1. INTRODUCTION

Mobile map applications are becoming increasingly popular on current Smartphones such as the Apple iPhone and the Android-based T-Mobile G1. This trend is likely to continue into the future, with the newest devices adding additional navigation-related features such as magnetic compasses.

We believe, however, that navigation interfaces for mobile maps can be significantly improved. Mobile maps provide expansive and simultaneously dense data on a wide range of scales, which makes displaying all data relevant to a specific navigation task difficult due to the small screen size on mobile devices. Using current mobile map interfaces, finding (and selecting) points of interest on mobile maps requires the user to perform numerous manual zooming and panning steps. This problem is exacerbated by the lack of a precise pointing device, such as the mouse on desktop PCs, on touch-screen-based mobile devices.

Speed-Dependent Automatic Zooming (SDAZ) [9] is a rate-based scrolling technique, in which the application automatically controls the content's zoom level depending on the scroll speed. The motivation behind automatic zooming is to preserve the optimal visual flow rate of the content by zooming out when the scroll rate increases. Also, switching between panning and zooming can increase mental load and user frustration. Thus, SDAZ is beneficial because it relieves users of an additional control dimension, the need to manually adjust the zoom level and allows simultaneous zooming and panning of the user interface. However, past studies have shown that SDAZ is not always comfortable for the users, although the results suggested that it improves rate-based scrolling [5]. As far as we know, no studies have yet been made that evaluate SDAZ interfaces for mobile maps which use 2D tilt input.

We examine if dynamic rate-based control with automatic zooming as implemented in SDAZ is potentially a beneficial technique for map navigation tasks in mobile map applications. However, for precise selection tasks (i.e. selecting a point of interest on the map), SDAZ can become uncomfortable to use, as users are forced to slow down to precisely pinpoint their target. This can precipitate imprecise increases in zoom at higher zoom levels, in which users "lose" their targets from the interface's view. Such conditions unnecessarily force the users to resume scrolling in order to regain an overview of their location by zooming out. This target over- or undershoot (or "hunting") problem [9, 16] significantly adds to task execution times when using SDAZ.

In this work we aim to show that exclusively coupling the content's zoom level to the scroll rate is an unfair constraint, and that especially in the case of mobile map interfaces, the user is likely to benefit from zooming out the content without having to scroll it, as map navigation tasks require users to view and to classify larger features of the map (in a zoomed-out state) in order to locate the point of interest they are interested in. Thus, we propose a novel SDAZ-based mobile map navigation technique, which we refer to as Semi-Automatic-Zooming (SAZ). The idea of SAZ is to use a tilt-based SDAZ interface for basic behavior, with added manual zoom level control, i.e. in the form of a slider widget mapped directly to the displayed content's zoom level. Using manual zoom control, users can keep the zoom level constant without the need to scroll the map. On the other hand, when scrolling to distant points of interest, users are relieved of the need to explicitly control the zoom level

The rest of this paper is structured as follows. In the next section, we discuss related work on interfaces with automatic zooming. Fol-

MobileHCI'10, September 7–10, 2010, Lisbon, Portugal.
ACM 978-1-60558-835-3/10/09.

lowing that, we present the implementation of our map interface, our implementation of SDAZ, as well as the novel SAZ interface and the multi-touch interface used for comparison in the user study. Thereafter, we discuss the user study in which we compared SDAZ, SAZ and multi-touch and present the results of our experiment. In the final section, we draw conclusions from this work and show how SAZ can be further improved, as well as pointing out directions for further research on the topic.

2. RELATED WORK

SDAZ is not a recent navigation technique, and was discussed by Igarashi et al. [9] in 2000 as a technique for navigating large documents. They conducted a preliminary user study comprising a 1D document scrolling task and a 2D map navigation task and compared SDAZ to traditional pan-and-zoom navigation. Whereas SDAZ was clearly preferred by the subjects in the 1D scrolling task, the preference for SDAZ in the map navigation was only slightly higher than traditional navigation. Moreover, the authors did not observe an improvement in task completion time using SDAZ in either the 1D or 2D tasks. Cockburn et al. conducted a similar study with a larger number of participants [4]. They, too, compared traditional with SDAZ navigation for 1D document scrolling and 2D map scrolling. Interestingly, their results are significantly in favor of SDAZ, both in terms of task completion times and NASA TLX [8] workload assessments. However, the authors used a 2D map display with strict boundaries, unlike modern "slippy" maps, as tested in our study and found, for example, in applications like Google Maps. Slippy maps allow for infinite scrolling in every dimension (of course with rollover at the map content boundaries). The map interface we used in our study contains enough map material to allow for zooming navigation over a substantial amount of zoom levels and geographic area. To our knowledge, such a realistic testbed has not been used in previous studies on mobile devices.

An evaluation of SDAZ scroll speeds for a 1D text scrolling interface was conducted by Wallace et al. [17]. Comfortable SDAZ scroll speeds are dependent on the assigned task (i.e. reading generally allows higher movement rates than looking at abstract data), visual perception, and the size and resolution of the device's display. We could not apply the results of that paper directly due to the different target domain (mobile map navigation as opposed to text document scrolling) and because their study was conducted using a 19 inch monitor connected to a desktop PC. We did however include some high-level practical advice, such as preferring an early zoom-out, in the interface for our study. Cockburn et al. conducted a similar study, comparing SDAZ and several variants of SDAZ (as proposed by Wijk et al. [15]) with traditional scroll bars in a 1D document scrolling task.

Murray-Smith et al. created a dynamic systems-based approach to model SDAZ [6, 7] and discussed its application in a tilt-based 1D text browsing interface for PDAs. In their approach, the the SDAZ "camera" is modelled as a physical object in an environment simulating mass and friction, which is coupled to the user's input. This model promises an automatic zooming behavior that is likely to be more easily understood by the users, as it directly follows physical analogies. Additionally, the dynamic systems-based approach allows developers to precisely tune the interface's behavior using only a limited set of parameters. In this contribution, we have adapted this previous work by extending the model proposed by the authors to enable 2D map scrolling with automatic zooming.

A paper with significant relevance to our paper is Appert et al. 's *OrthoZoom Scroller* [1], a 1D scrolling technique that uses one mouse axis to input the panning speed and the orthogonal axis to control the content's zoom (a similar technique, *GestureZoom*, had been previously presented by Patel et al. [13]). Appert's results have shown that OrthoZoom can be twice as fast as SDAZ. The zoom-level slider in our SAZ prototype interface has a similar function as the orthogonal axis in OrthoZoom and can also be considered an orthogonal input dimension. In contrast to OrthoZoom, the slider in our implementation does not totally override the automatic zooming behavior but only sets the base zoom level for SDAZ to operate on.

A study comparing pen-based rate control for map scrolling using SDAZ, pen-pressure-based zooming and tilt-based zooming was conducted by Büring et al. [3]. The authors conducted their study on a tablet PC with the content scaled to the screen resolutions of mobile devices. In contrast to our study, they did not use tilt for rate-control and their map content covered a much smaller geographic area, which is likely to have reduced the amount of zoom levels needed. In their study, the application's display size has a significant effect on the semantic (i.e. unguided) navigation tasks, SDAZ significantly reduces the task completion times in navigation tasks using HALO [2] to guide the users and a non-significant increase in task completion times when SDAZ is used in unguided navigation tasks.

3. IMPLEMENTATION

In order to study tilt-based mobile map interfaces, we implemented a custom mobile map application on the Apple iPhone 3GS. Our application has three navigation modes: a "standard" multi-touch based-map interface, a tilt-based SDAZ interface and an SAZ-based interface.

3.1 Slippy Map

We used the RouteMe toolkit [14] to implement the slippy map interface used in our test application. RouteMe allows the use of custom tile sources, and allowed us to generate a custom map tile database which was stored directly on the mobile device in the map application's folder.

The map tiles for the database were generated from OpenStreetMap [12] vector data, which is available for download at no cost. Our approach using an on-device map tile database has the advantage of removing any network latency due to map tile retrieval, and allows us to scroll and zoom the map dynamically at a maximum refresh rate of 20 fps.

The map tile database contains tiles for the world map from OpenStreetMap zoom levels 3 (19567.88 meters per pixel (m/px)) to 10 (152.87 m/px) and a detailed map of Central Europe is covered from zoom levels 11 (76.44 m/px) to 15 (4.78 m/px) which is detailed enough for identifying street names. In total, about 3.5 million map tiles are stored in the database, which has a size of almost 12 gigabytes.

3.2 Touch Interface

The multi-touch-based interface is modeled after the user interface of the iPhone's on-board map application. Scrolling is controlled by dragging the finger across the map. The pixel distance between the start and end of dragging is directly mapped to the map canvas. A drag motion of n pixels scrolls the map by n pixels in the specified direction. Zoom is controlled by two-finger gestures. A "pinch-in" gesture zooms out the map and a "spread-out" gesture zooms in the map.

Zooming is implemented by multiplying the current zoom level with the ratio of the starting and ending distance measured between the touch points of the pinch and spread gestures.

3.3 Speed-Dependent Automatic Zooming (SDAZ) Interface

In order to implement SAZ as proposed by us, we first had to implement a working tilt-based SDAZ map interface. Scroll rate control is achieved by tilting the device and using the resulting acceleration data as input.

3.3.1 State-Space Model for SDAZ

We chose to implement SDAZ using the dynamic systems approach as described in [7] because this is the current state of the art in SDAZ implementations and has been widely published. As the authors only specify a model appropriate for 1D scrolling tasks, we extended their model to be able to support 2D scrolling tasks with tilt input. To do this we had to extend the model update matrix to incorporate 2D input and movement, by adding two additional rows representing the scroll speed and position of the additional movement dimension:

$$\dot{X} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-R_h}{M} & 0 & 0 \\ 0 & 0 & 0 & \frac{-R_h}{M} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{A}{M} & 0 & 0 \\ 0 & \frac{A}{M} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \tag{1}$$

$\dot{X}$ represents the changes of the state-space vector with $x_1, \ldots, x_5$ representing *PositionX, PositionY, SpeedX, SpeedY* and *Zoom*. M, R_h and A are parameters of the state-space model, which we discuss in more detail in Section 3.3.2. M represents a mass in kg, R_h is the vertical resistance and A is a scaling constant. u_x, u_y are the tilt values obtained from the device's acceleration sensor, which is sampled at 20 Hz. The tilt values are used to update the state-space model as shown in Equation 1. The tilt values are computed using the arctangent of the acceleration values of the x (a_x) and y (a_y) axis divided by the acceleration value of the z (a_z) axis, respectively:

$$\begin{aligned} u_x &= atan(a_x/a_z) \\ u_y &= atan(a_y/a_z) \end{aligned} \tag{2}$$

Before calculating the tilt values, we applied a basic low-pass filter in order to decrease noise and lower the impact of unwanted motion. The low-pass filter however cannot have too much influence on the values because otherwise sudden corrective scrolling movements would be too sluggish. Because of the two-dimensional input (tilt in x and y axis) we also had to adapt the update of the zoom level of the system described by Murray-Smith. In Equation 1 the zoom update is represented by the u_z component. The zoom level update u_z is dependent upon the speed component sc and the current acceleration a_z in the z axis. We calculate u_z as follows:

$$u_z = (B/M)sc + (C/M)(1 + a_z) \tag{3}$$

B and C are scaling factors, and M is the mass value defined for the model. As device tilt increases, a_z decreases. When the device is completely level $a_z = -1$. On the iPhone, $a_z < 0$ when the screen faces upwards. An interesting property of Equation 3, is that we can adjust the influence of both tilt (a_Z) and speed (sc) on the interface zoom. This way, we can, for example, set up the interface zoom to respond faster to device tilt changes. This allows the user to get a better overview of the direction he wants to navigate to by increasing the device's tilt to zoom out, before a higher scroll rate gradually sets in.

To determine the speed component sc with which to update the zoom level, we look at the model ($\dot{x_3}$ or $\dot{x_4}$) to choose the component with the higher speed:

$$sc = max(abs(\dot{x}_3), abs(\dot{x}_4)) \tag{4}$$

3.3.2 Parameters of the State-Space Model

To allow for the camera to zoom continuously between zoom levels 3 and 15, and to fine-tune the coupling between scroll speed, tilt and zoom change, we had to adjust the parameters $M, R_h, A, B, C,$. In effect, these parameters model the physical properties of the virtual camera used in the SDAZ visualization. These properties are mass (M, in kg) and horizontal movement friction (R_h, in kg/s). A, B, C are used as scaling factors for the output values provided by the model.

In order to find values for the parameters, we sampled actual accelerometer data and ran an off-line simulation using Matlab. By analyzing plots of position, speed and location produced by the state-space model and different parameter settings, we found an initial set of values for the parameters. Later, the parameter values were tested by expert users in the map navigation interface implemented for our study. For map navigation, feedback provided by expert users suggested that we use model settings that make the map zoom out quickly upon tilt input. This enables the user to orientate himself more easily, such that she can find the correct movement direction towards his target point of interest. We had to slightly adjust the model parameters obtained via our offline analysis and used the following settings in our user study:

$$M = 50, R_h = 5, A = 1.5, B = 100, C = 30 \tag{5}$$

3.3.3 Visual Feedback and Diving Mode

Because SDAZ requires the user to center the screen on the exact location of the target point of interest before zooming in to see its details, we added a box around our interface's map center, as shown in Figure 1. In the following we will refer to this box as the "map center box".

When using SDAZ, it is crucial for the users to know exactly how they are influencing the interface's rate control, in order for them to predict its future behavior. In our SDAZ interface, rate control is achieved by tilting the mobile device. Because we found that users had difficulty to exactly judge the device's tilt, we implemented a visual feedback mechanism to visualize the current tilt angles. Visualization of the tilt in the x and y axis is achieved by drawing a small yellow square ("tilt indicator") in the area enclosed by the map center box (Figure 1 (a)) . If the device is level, the red dot remains in the center of this area, and indicates to the user that no scrolling is taking place. If the device is tilted in a certain direction, the red dot moves outwards from the center proportionally to the magnitude of the tilting, thus indicating the current scroll rate setting in the direction represented by the dot's offset with respect to the map center. To prevent operation of the interface at tilt angles larger than 30°, at which the map would not be clearly visible to the user anymore, we introduced feedback for an "over-tilt" mode, which freezes the interface and displays a warning rectangle around the map center box (Figure 1 (c)). When the device's tilt is within 10 percent of the boundary after which over-tilt is likely to occur ($\geq 27°$), the tilt indicator is enlarged and surrounded by a red bor-

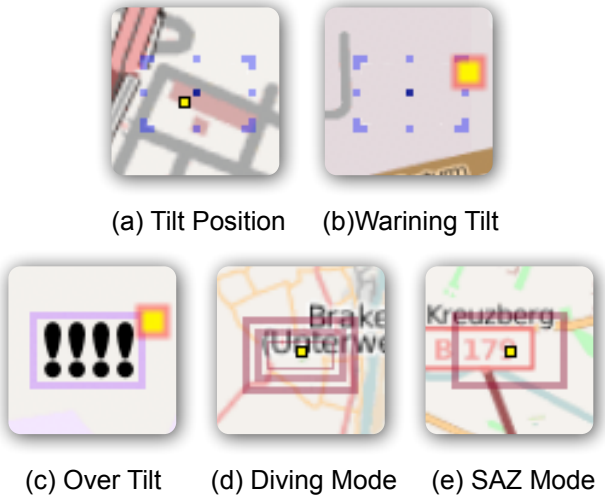

Figure 1: ***(Color required)*** **Visual feedback indicators for SDAZ and SAZ: (a) shows the visual feedback in default mode, with the tilt indicator displayed as a yellow square. (b) shows the warning-tilt indicator, when the tilt of the device approaches the over-tilt region. (c) indicates that the device has been tilted too far and that the interface is in over-tilt mode. (d) is the feedback displayed when in diving mode. (e) shows the default visual feedback given in SAZ mode.**

der, in order to warn the user that the over-tilt condition is imminent (Figure1 (b)).

A common problem with SDAZ interfaces is the overshooting of targets, or "hunting effect" [9, 16]. Because our interface simulates physical properties, such as friction and inertia, it is possible to overshoot the target, which makes the user compensate by tilting the device in the opposing direction. This, in turn increases the zoom level of the interface requiring further adjustment. A solution to this problem is to introduce a "diving mode" [7]. In our implementation, diving mode is initiated when the device is kept level for a time threshold of half a second. When diving mode is activated, the interface smoothly zooms the map to the maximum zoom level. The map center box changes its shape (Figure 1 (d)) to indicate to the user, that the interface is in diving mode.

3.4 Semi-Automatic Zooming (SAZ) Interface

The idea of Semi-Automatic Zooming is to give the user manual control over zooming when he desires it. In our case, this manual control is implemented as a slider (or "SAZ slider"). As shown in Figure 2, the SAZ slider is placed on the right-hand edge of the screen. This placement of the slider is optimized for use with a right-handed user's thumb, but could just as easily be designed for left-handed use by moving the slider to the left edge of the screen.

The slider's thumb is mapped linearly to the interface's zoom level, and automatically updates its position to reflect the current zoom level when manual zooming is not being performed by the user. The user can manually take control of the zoom level by touching the slider with his thumb (Figure 2 (b)). This is reflected in a change of color change of the slider area from transparent gray to magenta. The movement of the thumb is mapped in the following way: moving the slider up increases the zoom level, moving it down decreases the zoom level. Holding the slider's thumb at

Figure 2: Placement and visualization of the zoom level slider in the SAZ map interface: (a) shows the SAZ interface in automatic zooming mode (no touch on slider), (b) shows the manual zooming mode (finger on slider thumb controls the zoom level).

a constant position holds the zoom level constant. As the thumb's movement range is mapped directly to the range of zoom levels, the user can access all map zoom levels by a single movement with his thumb. Automatic zoom control is resumed when the the slider's thumb has been released by the user. Automatic zoom control is indicated by the slider's color changing from magenta to transparent gray (Figure 2 (a)).

Scroll rate control is not affected by use of the slider, and tilt input to scroll remains enabled during use of the SAZ slider. One of the major advantages of using an underlying model-based SDAZ interface is that we can keep the visual scroll speed constant irrespective of manual zoom changes input with the SAZ slider. Due to the availability of manual zoom control, dive-mode has been disabled in SAZ.

Several features of the visual feedback displayed in SDAZ mode are retained in SAZ mode. Here, the tilt indicator is surrounded by a red box (Figure 1 (e)). SAZ mode also shows over-tilt and over-tilt warning feedback.

4. USER STUDY

The aim of our experiment was to to compare task execution time, task load and user satisfaction results for the Touch, SDAZ and SAZ map interfaces. Additionally, we compared two different task types, *find landmark* and *follow route* . Each task type is divided in two sub tasks with two different map scales *large* (regional to country scale) and *small* (city scale).

4.1 Participants

We invited a total of 13 right-handed participants, 8 male and 5 female. The average age was between 24 and 28 years. The major-

(a) Landmark Marker (b) Route Markers (c) Route Segment Update

Figure 3: (a) shows one of the markers as displayed in the *find landmark* task. (b) shows the markers to be crossed in the *follow route* task. (c) shows how the last route segment is updated (color change from yellow to orange) when the user crosses into the next route segment in the *follow route* task.

ity of the participants did not have prior experience with either the iPhone or mobile map navigation.

Participants were given monetary compensation after completing the study.

4.2 Experimental Design and Tasks

The experiment used a repeated measures, within-participant factorial $3 \times 2 \times 2$ design. Factors were input method (*multi-touch, SDAZ, SAZ*), task type (*find landmark, follow route*) and map scale (*large, small*).

The goal of the *find landmark* task was to analyze the usefulness of the interface control mode for searching for the exact location of a point of interest. The *find landmark* task was designed to represent finding the exact location of a certain point of interest, such as a restaurant, train station or popular sightseeing destination in an actual mobile map application. In a deployed mobile map application, the *follow route* task would be analogous to following a subway line map, for example.

In *find landmark* the task of the user was to locate and select a number of landmarks by navigating to them and selecting them by tapping on them. The landmark markers (Figure 3 (a)) were only visible from zoom level 11 onwards in the *large* map types and from zoom level 13 in the *small* map types, due to the smaller geographic area which resulted in a lower need to zoom out to lower zoom levels. Selection of the landmarks was only possible when fully zoomed in. The marker display constraints for the *find landmark* task were implemented to force the users to precisely locate and select the landmarks. During the trials, the participants were given a paper overview map on which the location and sequence of the landmarks for the current trial was shown.

The task in *follow route* was to follow a given route using the mobile interface. Each route consists of a number of waypoints (Figure 3 (b)) on that had to be crossed in sequence. The waypoints are connected by the route shown in yellow. When a waypoint is crossed, the color of the previous segment of the route is changed to red, to indicate the direction of the next waypoint, as is shown in Figure 3 (c). Waypoints could only be crossed from zoom levels 13 to 15. The aim of this was to make the test subject follow the route as closely as possible.

Each task was tested on large and small map scales. Our idea was that regional-scale (150-500km map diameter) map areas would force the user to zoom out to a lower zoom level as compared to the same task on a city-sized map scale (25-100km map diameter). By having two types of map scale, we could measure the performance of the input techniques over a wider range of zoom levels and geographic areas. The *follow route* task was designed to evaluate the panning functionality whereas in the *landmark finder* task participants had to make additional use of the zooming functionality [3].

4.3 Apparatus

We evaluated SAZ, SDAZ and multi-touch using a custom-built application running on an iPhone 3GS. In order to prevent lag from loading map tiles, all map tiles were stored in a database on the device. The map data contained the entire world map from zoom levels 3 to 10 (zoom levels 1 and 2 were not used) and a detailed map (zoom levels 11 to 15) covered the boundaries of Central Europe.

For the *find landmark* task, we provided the users with a printed paper map, on which the the location of the landmarks was indicated. We chose to provide a paper map in order to disambiguate our results from the participants' geographic knowledge.

4.4 Procedure and Dependent Measures

The study began with the participants filling out a general questionnaire on their gender, age and experience with smart phones and mobile maps.

As a warm-up exercise, the participants were required to play three levels on a tilt-based labyrinth game [10]. The goal of this exercise was to help the user learn the effects of tilting the device when using a tilt-based interface. The Labyrinth program gave the users an indication of the lag (if any), sensitivity and precision of the built-in acceleration sensor. Furthermore, the movement of the ball when the device was tilted was analogous to the movement direction of the camera in the SDAZ and SAZ interfaces. This may have helped to adjust the users' mental model when using the SAZ and SDAZ map interfaces in the actual user study.

We counterbalanced the order of input technique to prevent learning effects. For each input technique, we tested two task variants, *find landmark* and *follow route* as well as two map scales (large, small) per subtask, which in total resulted in 12 trials (one trial per condition) per user.

We measured task completion time as a quantitative performance measure.Task duration was measured directly on the test apparatus. Task duration gives us a good indication of the performance of the input technique, although as can (also) be seen from our results, task duration does not always correlate with satisfaction of ease of use.

Further Qualitative measures were *task load* (measured using the NASA TLX questionnaire [8]) by input method and *satisfac-*

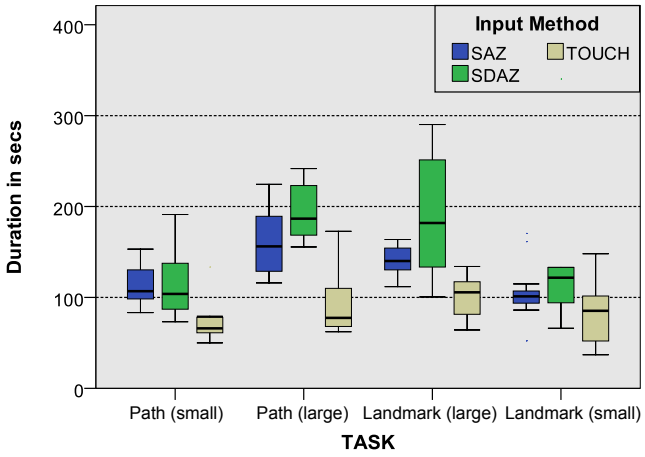

Figure 4: Boxplot of task completion time by input method and subtask.

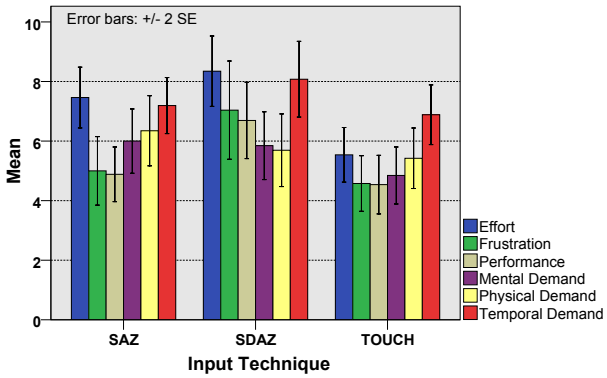

Figure 5: Results of NASA TLX questionnaires by input technique.

tion, learnability, usefulness and *ease of use* (measured with the USE questionnaire [11]) and their respective ranking for each input method.

A NASA TLX questionnaire was provided to the test subjects after each trial, to record the direct impact each test condition had on the user's task load. Users were required to fill out the USE questionnaires after completing all the trials for each input method. Apart from the USE questions, participants were asked to provide negative or positive comments about the input technique, if they had any. After completion of the study, the users had to give a ranking of the USE variables on a further questionnaire.

4.5 Results

4.5.1 Task Completion Time

Subtask	Input Technique	Mean (s)	Std. Dev. (s)
Path (Small)			
	SAZ	113.1	21.5
	SDAZ	117.4	35.9
	Multi-Touch	74.3	24.6
Path (Large)			
	SAZ	163.3	37.0
	SDAZ	200.4	42.7
	Multi-Touch	91.5	32.6
Landmark (small)			
	SAZ	106.8	30.3
	SDAZ	139.8	75.3
	Multi-Touch	83.9	37.5
Landmark (large)			
	SAZ	140.2	27.4
	SDAZ	192.3	65.0
	Multi-Touch	100.1	22.9

Table 1: Average task completion times by subtask and input technique.

Table 1 shows the average task completion times ordered by subtask and input technique. The task completion time of SAZ is lower than SDAZ in all subtasks, and multi-touch was the faster technique across all subtasks.

Figure 4 shows the boxplots of the task duration vs. input method and subtask. A univariate ANOVA yields a combined significant effect of input method $\times$ task type on the task duration ($F(5, 155) = 13,756, p < 0.001$). However task type (*find landmark* or *follow route*) alone did not have a significant effect ($F(1, 155) = 0.004, p = 0.948$). In contrast, input method showed a significant effect on task completion ($F(2, 155) = 33.363, p < 0.001$). There was no significant input method $\times$ task type interaction.

4.5.2 NASA TLX

To analyze the effects of input method(multi-touch, SAZ or SDAZ) and task type (*find landmark* or *follow route*) on the NASA TLX ratings of *Effort, Frustration, Performance, Mental Demand, Physical Demand* and *Temporal Demand*, we conducted a multivariate ANOVA.

The following measures showed a significant difference:

- *Effort:* $F(5, 150) = 3.27, p = 0.008$
- *Performance:* $F(5, 150) = 2.64, p = 0.03$
- *Mental Demand:* $F(5, 150) = 2,95, p = 0.01$

Input technique had a significant individual effect on the following measures:

- *Effort:* $F(2, 150) = 7.48, p < 0.001$
- *Frustration:* $F(2, 150) = 4.19, p = 0.02$
- *Performance:* $F(2, 150) = 4.68, p = 0.01$

There was no significant interaction of input method $\times$ task type.

The effects of different input techniques on *Effort, Frustration* and *Performance* are as expected, due to the different characteristics of the input techniques. Mental, physical and temporal demand do not appear to be significantly affected by the input technique. The sole significant individual effect of task type was *Mental Demand* ($F(1, 150), = 9.611, p = 0.002$), which can be explained by the difference in difficulty of our tasks. *Follow route* appears to be significantly easier than locating and finding a target on the mobile map, even with the target marked on a paper map for guidance.

To analyze the individual differences of the the input techniques, we conducted a Sidak post-hoc analysis. For *Effort* there is a significant difference between multi-touch and SAZ ($MD = 1.92, p = 0.032$) as well as multi-touch and SDAZ ($MD = -2.81, p = 0.001$), although there is no significant difference between SDAZ and SAZ. For *Frustration*, multi-touch differs significantly from SDAZ ($MD = -2.46, p = 0.023$), whereas there is no significant difference between SAZ and multi-touch. There is a borderline significant difference in *Performance* between SAZ and SDAZ

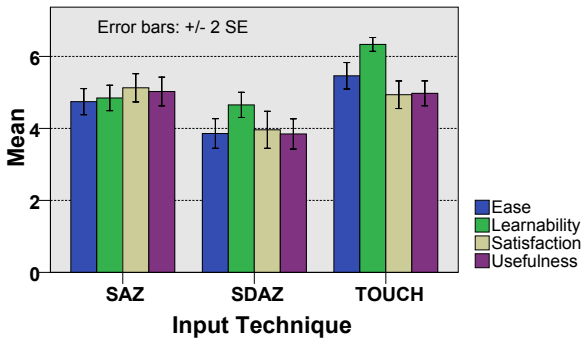

Figure 6: Results of USE Questionnaire by input technique (higher is better).

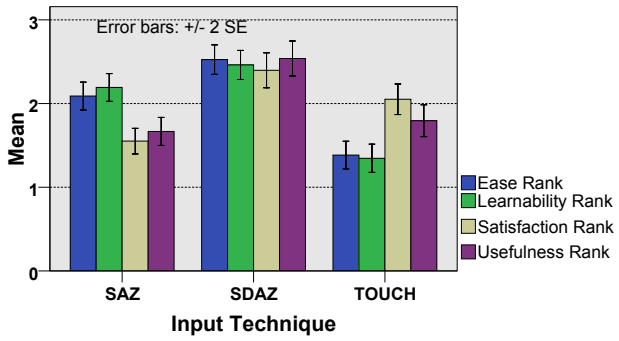

Figure 7: USE Questionnaire ranking results.

($MD = -1.81, p = 0.053$) and between multi-touch and SDAZ ($MD = -2.15, p = 0.15$). Interestingly, there is no significant difference in *Performance* between SAZ and SDAZ. Conforming to the results of the multivariate ANOVA, *Mental Demand, Physical Demand* and *Temporal Demand* had no significant effects.

Figure 5 shows a plot of the average ratings of the NASA TLX measures by input technique. In terms of user-rated *frustration*, there was no significant difference between SAZ ($M = 5.0, SD = 4.14$) and multi-touch ($M = 4.58, SD = 3.36$). SDAZ was rated significantly higher in *frustration* with the worst average rating of 7.04 ($SD = 5.94$). The user rated the *effort* worst for SDAZ ($M = 8.35, SD = 4.26$) with no significant difference compared to SAZ ($M = 6.44, SD = 3.68$). Multi-touch was rated significantly better for *effort* with an average of 5.54 ($SD = 3.3$) A notable NASA TLX result is the (borderline) significant increase in user-rated *performance* of SAZ over SDAZ ($M = 4.88, SD = 3.3$ vs. $M = 6.69, SD = 4.6$; lower is better), however multi-touch remains best ($M = 4.54, SD = 3.55$).

4.5.3 USE Questionnaire

The mean ratings given for each input technique on the USE Questionnaire are shown in Figure 6. We conducted a multivariate ANOVA on the results of the USE Questionnaire. Dependent measures were *Ease, Learnability, Satisfaction* and *Usefulness*. Task type and input method were chosen as factors. There were significant differences between all measures. The individiual F-values are as follows:

- *Ease:* $F(2, 155) = 7.214, p < 0.001$
- *Learnability:* $F(2, 155) = 14.097, p < 0.001$
- *Satisfaction:* $F(2, 155) = 3.286, p = 0.008$
- *Usefulness:* $F(2, 155) = 4.839, p < 0.001$.

Task type had no significant effect on any of the measures whereas input technique had a significant effect on all measures:

- *Ease:* $F(2, 155) = 17.506, p < 0.001$
- *Learnability:* $F(2, 155) = 35.131, p < 0.001$
- *Satisfaction:* $F(2, 155) = 8.157, p < 0.001$
- *Usefulness:* $F(2, 155) = 11.567, p < 0.001$.

4.5.4 USE Rankings

Figure 7 shows the results of the USE questionnaire rankings. Sidak post-hoc analysis for the USE Questionnaire rankings shows that for *Satisfaction* and *Usefulness*, SAZ ranks at least as well as multi-touch. For the other USE measures, SAZ consistently received a higher rating than SDAZ and SAZ was rated significantly higher than SDAZ in *Ease*. The difference in *Learnability* between SDAZ and SAZ was not significant, as these two techniques seem harder for unskilled users to master than multi-touch.

The detailed results for the Sidak analysis are as follows: for *Ease* there were significant differences between SAZ and SDAZ ($MD = 0.88, p = 0.004$), between multi-touch and SAZ ($MD = 0.72, p = 0.027$) and also between touch and SDAZ ($MD = 0.88, p < 0.001$). Multi-touch was rated best ($M = 6.56, SD = 1.48$), followed by SAZ ($M = 4.74, SD = 1.31$) and SDAZ ($M = 3.85, SD = 1.48$). *Learnability* had the highest rating for multi-touch ($M = 6.34, SD = 0.69$) with a significant difference compared to SAZ ($MD = 1.49, p < 0.001$) and SDAZ ($MD = 1.68, p < 0.001$). SAZ had the second best rating ($M = 4.85 SD = 1.27$), with SDAZ worst ($M = 4.65, SD = 1.26$). There was no significant difference in *Learnability* between SAZ and SDAZ. SAZ had the best rating for *Satisfaction* ($M = 5.13, SD = 1.42$), with multi-touch second ($M = 4.93, SD = 1.38$), although the difference was not significant. SDAZ had the lowest rating ($M = 3.96, SD = 1.85$) with significant differences compared to both SAZ ($MD = 1.67, p = 0.001$) and multi-touch ($MD = 0.97, p = 0.006$). SAZ also had the highest rating for usefulness ($M = 5.03, SD = 1.44$) followed by multi-touch ($M = 4.97, SD = 1.25$), although also in the case the differences were not significant. SDAZ received the lowest rating ($M = 3.85, SD = 1.51$) that was significantly different compared to both SAZ ($MD = 1.18, p < 0.001$) and multi-touch ($MD = 1.13, p < 0.001$).

Figure 7 reflects clearly the borderline differences in *Satisfaction* and *Usefulness* rankings for the input techniques. On average, SAZ ranks first in *Satisfaction* ($M = 1.55, SD = 0.55$) and *Usefulness* ($M = 1.67, SD = 0.60$), with multi-touch (*Satisfaction:* $M = 2.05, SD = 0.67$, *Usefulness:* $M = 1.79, SD = 0.69$) second and SDAZ third (*Satisfaction:* $M = 2.39, SD = 0.55$, *Usefulness:* $M = 2.53, SD = 0.60$) . Multi-touch ranks first in *Ease* ($M = 1.38, SD = 0.60$) and *Learnability* ($M = 1.34, SD = 0.55$), with SAZ (*Ease:* $M = 2.089, SD = 0.60$, *Learnability:* $M = 2.19, SD = 0.60$) second and SDAZ third (*Ease:* $M = 2.52, SD = 0.63$, *Learnability:* $M = 2.46, SD = 0.63$).

The low *Ease* and *Learnability* ratings received by SAZ, which are not significantly different than those of SDAZ, indicate that this

technique is harder to learn for novice users. However, SAZ was always rated better than SDAZ and the ratings for *Satisfaction* and *Usefulness* indicate that the users perceived SAZ as being more satisfying to use and more useful than multi-touch. We believe that the ratings for *Satisfaction* and *Usefulness* would have been even more favored towards SAZ if the users had had more experience with that technique.

4.6 User Feedback

At the end of the USE Questionnaires for each input method, the users were asked to give positive or negative feedback, if they felt inclined to do so.

In the case of SDAZ, some users complained about "unwanted loss of control" due to the coupling of tilt with zooming and scrolling. However, users also noted some positive aspects of SDAZ, such as "map always visible" (i.e. no occlusion), "quick zooming" and "fun" to use.

Regarding multi-touch, the users liked the "precise control" they had over the map view. Frequent negative issues with multi-touch were "screen is covered" (i.e. occlusion), "slow" and "sticky fingers". It is interesting to note that multi-touch was perceived as being slower although this is contradicted by the experimental results. This perception might be due to the mental load associated with switching between zooming and panning the interface (both is not possible simultaneously using multi-touch).

The users liked the "quick zooming" of SAZ and found this input method "fun to use". They also noted that, in contrast to multi-touch, SAZ makes "one-handed interaction possible". However, the users recommended the slider to be designed in a way that it can be "locked" at a certain zoom position without the requirement of maintaing touch contact with the slider. Also, the users commented that they would "get better with more practice" when using the SAZ interface.

The user feedback we obtained indicates that although multi-touch had the best experimental results, some users did perceive multi-touch as being a slow technique and had the impression of being faster while using SAZ. Feedback provided by the users also suggests that of the main advantages of using an SDAZ or SAZ interface is that these techniques are occlusion-free, which was appreciated by many of the users. SAZ was generally preferred over SDAZ due to the availability of manual zoom control.

5. CONCLUSION

In order to conduct our study comparing SDAZ, SAZ and multi-touch for mobile map navigation, we had to implement a high-performance dynamic map interface. Because we wanted to conduct our study under the most realistic conditions possible, we chose to implement our interfaces directly on a mobile device, and to use actual real map material at a high detail level covering a large geographic area with a wide range of scales. This has not been achieved in previous work in the field.

Rather than choosing an arbitrary mapping between scroll rate and zoom level, we chose to use the state-space model by Murray-Smith [7], as we think that this model provides a more natural mapping. We had to substantially extend the existing model to enable the support of 2D tilt input for scroll rate control.

Our study demonstrates that for mobile map navigation, SAZ is a superior input technique compared to SDAZ. Not only was the task duration of SAZ significantly lower, SAZ was rated better in both the NASA TLX and the USE questionnaires. Notably, the results from NASA TLX and the USE Questionnaire indicate that SAZ performs at least as well as multi-touch. The user feedback and *Satisfaction* and *Usefulness* ratings from the USE Questionnaire indicate that SAZ was seen as increasing the user's productivity and effectiveness while at the same time being fun to use. A possible reason that the study results are not even more in favor of SAZ is that the users had no previous experience with automatic zooming or tilt-based interfaces. Also, we have to take note that the iPhone's support for multi-touch is very mature, as this device was developed for supporting multi-touch as primary means of interaction. This may have contributed towards biasing the study results towards multi-touch. In this context, it is interesting to note that multi-touch was perceived as being the slowest technique by a number of users. Additionally, the users mentioned occlusion problems when using the multi-touch interface, which weren't present when using SDAZ or SAZ. Because of the very mature implementation of multi-touch on the iPhone, it may be useful to re-evaluate SAZ and SDAZ in comparison to a non multi-touch mobile map interface such as the one found on mobile devices running Android. In this case, SAZ has the potential to be at a clear advantage.

From a more global perspective, the results of this work demonstrate that tilt-based SAZ is an efficient input technique for navigating large information spaces on mobile devices, and has the potential to be a very useful input technique for future devices that are not equipped with multi-touch-based interfaces.

6. FUTURE WORK

From the user feedback, we gained some valuable insight into the design of future SAZ interfaces. The users appreciated the manual zoom control provided by the SAZ slider, although it was apparently lacking a feature allowing the users to lock the zoom level without having to touch its thumb.

In future work, it may be useful to analyze how much the users made use of the SAZ slider in individual task scenarios, i.e. in *find landmark* or *follow route*. In addition, we believe that in certain scenarios involving a search for a specific point of interest, such as in the *find landmark* task in our study, there may be certain phases of the task in which the users prefer to use automatic zooming and other phases where manual zoom control is preferred. We wish to analyze the phases of SAZ slider usage in more detail in future user studies.

7. REFERENCES

[1] C. Appert and J. D. Fekete. OrthoZoom scroller: 1D multi-scale navigation. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 21–30, 2006.

[2] P. Baudisch and R. Rosenholtz. Halo: a technique for visualizing off-screen objects. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 481–488, Ft. Lauderdale, Florida, USA, 2003. ACM.

[3] T. Büring, J. Gerken, and H. Reiterer. Zoom interaction design for pen-operated portable devices. *International Journal of Human-Computer Studies*, 66(8):605–627, 2008.

[4] A. Cockburn and J. Savage. Comparing speed-dependent automatic zooming with traditional scroll, pan and zoom methods. *PEOPLE AND COMPUTERS*, pages 87–102, 2004.

[5] A. Cockburn, J. Savage, and A. Wallace. Tuning and testing scrolling interfaces that automatically zoom. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 71–80, 2005.

[6] P. Eslambolchilar and R. Murray-Smith. Tilt-Based automatic zooming and scaling in mobile Devices-A

State-Space implementation. *Lecture Notes in Computer Science*, pages 120–131, 2004.

[7] P. Eslambolchilar and R. Murray-Smith. Control centric approach in designing scrolling and zooming user interfaces. *International Journal of Human-Computer Studies*, 66(12):838–856, 2008.

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

[9] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, San Diego, California, United States, 2000. ACM.

[10] IllusionLabs. Labyrinth Game for iPhone. `http://labyrinth.codify.se`, Accessed 23.09.2009.

[11] A. M. Lund. The need for a standardized set of usability metrics. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 42, pages 688–691, 1998.

[12] OpenStreetMap. OpenStreetMap Project. `http://www.openstreetmap.org/`, Accessed 23.09.2009.

[13] D. Patel, G. Marsden, S. Jones, and M. Jones. An evaluation of techniques for browsing photograph collections on small displays. *Mobile Human-Computer Interaction–MobileHCI*, pages 132–143, 2004.

[14] RouteMe. RouteMe Toolkit for iPhone. `http://code.google.com/p/route-me`, Accessed 23.09.2009.

[15] J. J. van Wijk and W. A. Nuij. A model for smooth viewing and navigation of large 2 d information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):447–458, 2004.

[16] A. Wallace. The calibration and optimisation of Speed-Dependent automatic zooming. *University of Canterbury, Christchurch, New Zealand November*, 2003.

[17] A. Wallace, J. Savage, and A. Cockburn. Rapid visual flow: how fast is too fast? In *Proceedings of the fifth conference on Australasian user interface-Volume 28*, pages 117–122, 2004.