An Evaluation Of Integrated Zooming and Scrolling On Small-Screens

Steve Jones  Matt Jones  Gary Marsden  Dynal Patel  Andy Cockburn
Department of Computer Science  Department of Computer Science  Dept of Computer Science  
University of Waikato  University of Cape Town  University of Canterbury  
Private Bag 3105  Private Bag Rondebosch 7701  Christchurch  
Hamilton, New Zealand  Cape Town, South Africa  New Zealand  
{stevej, mattj}@cs.waikato.ac.nz  {gaz, dpatel}@cs.uct.ac.nz  andy@cosc.canterbury.ac.nz

Abstract

Navigation within two-dimensional information spaces such as documents, images and spreadsheets is commonly supported by a combination of discrete scrolling, panning and zooming facilities. However, the number of navigational actions increases as the proportion of the space that is visible on a display decreases. The problem of increased navigational effort is exacerbated when using small screen display devices, which can display only small portions of the original space. An alternative navigation technique—speed-dependent automatic zooming—has been proposed for standard desktop displays. It replaces normative facilities by combining zooming and panning of the information space into a single operation, with the magnitude of both factors dependent on simple user interaction. In this paper we propose algorithmic extensions to this technique for application on small-screen devices, and describe an implementation that runs efficiently on a range of computing devices without the need for special hardware or software. We conducted a comparative experimental evaluation of user performance with the system and a normative interface. We found that XXX.

Introduction

Users deal daily with information spaces that are too large to be fully displayed within their available window, or even within a single screen. Documents, web pages, pictures, spreadsheets and filestore folders are types of information space that commonly suffer from this problem. A well-established technique for allowing navigation around large spaces is to provide a ‘viewport’ within which a subset of the space is displayed. The subset shown within the viewport can be controlled by the user, who conceptually moves either the viewport around on top of the space, or the space around under the viewport. Scrollbars are a common control mechanism for this interaction, supporting both continuous and discrete (often in terms of pages) navigation actions, with one scrollbar providing vertical viewport control and another providing horizontal control. A further mechanism allows the user to drag the space within the viewport (often termed panning), manipulating its location directly in any direction without constraint to either horizontal or vertical movements. Another approach increases or decreases the size of the information space subset visible in the viewport via a zoom function.

Although many systems provide all three of these scrolling, panning and zooming operations, there are numerous limitations to the navigation support that they provide. Igarashi and Hinckley [4] note the attentional overhead incurred in changing focus between document content and scrollbars. Users must initially focus attention on a scrollbar to situate the cursor within the appropriate control item—usually one of two directional arrows, a scroll handle or either side of the scroll handle. During a scroll operation the user must then consider both the effect of the operation on the document and further possible actions in the scrollbar. Igarashi and Hinckley [4] suggest that this can increase operational time. They also observe that small scrollbar movements can result in large movements of the viewport for long documents, causing disorientation and confusion for the user.

Cockburn and Savage [2] note that because zooming changes the visible proportion of the information space, more scrolling is required when zoomed in, and less when zoomed out to achieve the same
transformation of the viewport. Given that scrolling operations are dependent upon the current zoom-level, scrollbars can have varying effects in response to the small set of available user actions. To predict, or interpret scrollbar interactions, users must therefore understand the relationship between scroll-distance and zoom-level, adding further overhead to scrolling operations.

Users may be able to manipulate the zoom-level prior to scrolling or panning. For example, they may zoom out from the document (displaying more of it in the viewport), reducing the amount of subsequent scrolling activity because more of the document passes within the viewport with each scrolling action. The zoom-level can then be reset. However, this requires a number of interface actions, introducing further overhead for what is a very common activity. It is likely then that users incur higher scrolling costs as a trade-off for immediacy in manipulating the document.

When panning is not supported by an application, users are restricted to independent vertical and horizontal manipulation of the viewport. Consequently, navigation in other directions (such as diagonally) requires at least two scrolling operations to render a target location visible within the workspace.

The limitations of conventional scrolling techniques are of greater concern in the context of small-screen devices, such as Personal Digital Assistants (PDAs), Pocket PC, mobile telephones and some laptop computers. These devices provide severely limited display areas in comparison to standard desktop displays devices, possibly restricting the visible portion of an information space to a few percent of its overall area. Therefore, more scrolling is required to navigate within information spaces, with a likely commensurate negative effect on user interaction. Also, the presence of the scrollbars themselves requires valuable display area, further reducing the subset of the information space that can be displayed.

**Error! Reference source not found.** shows the display sizes of a selection of current handheld, portable and desktop devices. The distinction between the functionality and applications provided on handheld devices and their larger desktop counterparts is becoming increasingly blurred. Device B, for example, provides a Microsoft Windows environment with standard productivity tools such as Microsoft Word, Excel and PowerPoint, yet has only 4% of the display area of a 21-inch (diagonal) desktop screen (device F). Table 2 summarises the relative display areas of each of the devices. Device display areas are compared by reading across a device row to a device column. For example, a Compaq iPAQ 5400 provides 7% of the display area provided by a Titanium Powerbook 15”.

Some information objects, such as electronic mail messages, may wrap to the available display width, increasing scrolling in the vertical dimension only. Such re-formatting may not be appropriate or possible for other objects such as word-processed documents, spreadsheets or images. There is clearly a need then for a navigation interface mechanism that overcomes current scrolling limitations.
which the scrolled object moves causes visual blurring and consequent user disorientation. Some algorithm was applied. For these standard the standard for these devices, rate-based scrolling has an attendant problem, noted by both Igarashi and Hinckley [4] and Cockburn and Savage [2]. As the scroll-rate increases, the rapidity with which the scrolled object moves causes visual blurring and consequent user disorientation.

<table>
<thead>
<tr>
<th>Palm Tungsten T</th>
<th>Compaq iPAQ 5400 series</th>
<th>Nokia Communicator 9290</th>
<th>Nokia 6800</th>
<th>Apple Titanium Powerbook 15&quot;</th>
<th>Philips 202P monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>78</td>
<td>88</td>
<td>367</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>129</td>
<td>100</td>
<td>113</td>
<td>474</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>114</td>
<td>88</td>
<td>100</td>
<td>418</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>21</td>
<td>24</td>
<td>100</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1815</td>
<td>1407</td>
<td>1595</td>
<td>6667</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>3629</td>
<td>2813</td>
<td>3189</td>
<td>13333</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2: comparative display areas of a selection of common computing devices

In this paper we investigate the utility of an alternative navigation mechanism—speed-dependent automatic zooming—in the small screen context. This technique has been shown to improve user performance in navigation tasks, but has been evaluated on standard desktop displays only [2]. We have extended the underlying algorithm for this technique for application on displays of any size, even allowing for dynamic resizing of the application window by the user. We have carried out an empirical investigation of the utility of the resulting interaction.

In the following section of this paper we describe alternatives to standard scrolling mechanisms, focussing on the speed-dependent automatic zooming technique, and report on prior evaluations of its utility. We then describe our algorithmic extensions and our implementation that can be deployed on a range of devices. In the following two sections we present and report on a user study that investigated how well this technique supported users in navigation tasks on a small-screen device, and compare its efficacy to a standard navigation interface. Finally, we discuss the observed results and offer conclusions regarding the application of this technique to small-screen user interfaces.

**Related work**

One approach to ameliorating the limitations of conventional scrolling is to provide alternative input mechanisms, particularly via redesign of a pointing device such as a mouse. The Microsoft IntelliMouse exemplifies pointing devices that contain a ‘scroll wheel’. The wheel is rotated forward or backward by the user to control upward or downward scrolling, and is free-moving. The relationship between scroll distance and the magnitude of wheel movement can be adjusted to increase or decrease scroll speed, but scrolling only occurs in response to rotations of the wheel. However, the IntelliMouse is also a rate control device. When the wheel is depressed the degree of rotation controls scroll speed—the further that the wheel is rotated from its original position, the faster the speed of scrolling. When the wheel is released scrolling stops.

The IBM ScrollPoint mouse replaces the scroll wheel with an isometric joystick. The degree of pressure exerted on the joystick is mapped to scroll-speed. Zhai and Smith [5] carried out an evaluation of scrolling techniques, comparing these two styles of device, standard mouse-controlled scrollbars, and two-handed interaction via a standard two-button mouse and associated keyboard-located joystick. For navigation-based tasks, they found no significant difference between completion times for the standard mouse and mouse wheel, or between the joystick mouse and two-handed input. Under both the joystick mouse and two-handed input conditions subjects were significantly faster than with the standard or mouse wheel devices. Hinckley et al [3] provide further insight into the performance of these devices. They observed a crossover effect, with a mouse wheel supporting the best performance for short scroll distances, and a mouse-based joystick performing best for long distances. They also found that the mouse wheel performance could be significantly improved when an acceleration algorithm was applied.

Some small screen devices have isotonic or isometric wheel controls or joysticks that can be used for scrolling, and therefore go some way towards addressing scrolling issues. Although this may alleviate some effort for the users of these devices, rate-based scrolling has an attendant problem, noted by both Igarashi and Hinckley [4] and Cockburn and Savage [2]. As the scroll-rate increases, the rapidity with which the scrolled object moves causes visual blurring and consequent user disorientation.
Speed-dependent automatic zooming (SDAZ) was proposed by Igarashi and Hinckley [4] to alleviate blurring and other problems with conventional scrolling techniques. SDAZ links scroll-speed and zooming of the information space. As scroll-speed increases, the zoom-level decreases in parallel; that is, more of the information space is visible within the display area. For slow scroll-speeds or when no scrolling is occurring, the zoom level is set to its normal value. No scrollbars are required, and scrolling may occur in any direction, although it may be constrained to suit particular applications or types of information space.

SDAZ is initiated when the user presses the mouse button. Scrolling begins when the user drags the mouse away from the location where the mouse button was pressed, and the direction of the mouse in relation to its original position determines the direction of scrolling. The information space scrolls in the opposite direction to that of the mouse—the user is indicating the direction in which they wish to navigate. The scroll speed is proportional to the distance of the mouse from its original position, which is controlled by the user. The zoom-level is adjusted automatically during scrolling to reduce the effect of high visual flow and the consequent blurring that occurs when the document is scrolled quickly at normal scale.

Igarashi and Hinckley [4] initially suggest the following equations to compute speed and scale:

\[
speed = C \times dy
\]

**Equation 1**

\[
scale = s0(\frac{dy}{d0}) \times (d1/d0)
\]

**Equation 2**

In Equation 1 \( dy \) is the current distance that the mouse has been dragged (for vertical scrolling only). \( C \) is a constant that can be used to modify scroll speed, although suitable values are not reported. In Equation 2 \( s0 \) specifies the minimum scale to which the scrolled object can be zoomed. \( d0 \) provides a distance threshold below which zooming does not take place. This allows the user to scroll slowly within the distance specified by \( d0 \) without any scaling. \( d1 \) specifies the mouse distance beyond which no further zooming takes place. However, to achieve a more natural interaction they amend Equation 1 as follows

\[
speed = \frac{\nu0}{scale}
\]

**Equation 3**

where \( \nu0 \) is a constant defining the speed reached at \( d0 \), ie the speed up to which scroll speed in linearly related to distance. This amendment conflicts with the notion that zooming (or scale) is dependent upon scroll speed. In fact, the zoom level is dependent upon the distance that the user moves the mouse, and speed is dependent upon the resulting zoom level. Figure 1 replicates Igarashi and Hinckley’s graph of the behaviours of Equation 2 and Equation 3. Although the graph reflects the desired scrolling and scaling behaviour it does not accurately reflect the effect of the equations. Although the scale value is 1 when the mouse distance \( = d0 \), for values of \( dy \) greater than zero and less than 1, the scale value is greater than 1. Also, neither scale nor speed become constant at \( d1 \). Both of these effects must be produced programmatically,
Igarashi and Hinckley further modified their equations during implementation by introducing a delay in scaling. This was necessary to avoid instantaneous zooming to full size when the mouse button was released, and undesirable ‘swelling’ effects when the user reversed scrolling direction. This implementation was then evaluated in an informal usability study of two interfaces. The first interface was a simple web-browser, although the study tasks did not require users to navigate between web pages; in effect it was a vertically scrolling document browser. Subjects were required to locate target images within a web page, using standard vertical scrollbars and the SDAZ implementation. Task completion times were approximately equal, although six of the seven subjects preferred using the SDAZ interface. The second interface was a map browser, for which the maps were artificially generated. Subjects used a joystick to navigate to a marked location by panning across the map. One condition provided zooming via a joystick button, and the other via the SDAZ implementation. The task completion times were too diverse to provide evidence regarding the interaction modes, and subjective preferences were split four to three for the two modes (the majority preferring the SDAZ mode).

Cockburn and Savage [2] carried out a more substantial evaluation of their own implementations of SDAZ document and map viewing applications. Igarashi and Hinckley made some implementation compromises to achieve rapid interaction in their Java implementation, including rendering of small text as horizontal line placeholders. Cockburn and Savage created C/OpenGL implementations providing smooth animation at high frame rates. They used the following equation to compute the scrolling rate for the vertically scrolling document browser:

$$\text{scrollspeed} = | Y_{ip} \Delta Y \text{cp} |$$

Equation 4

where $Y_{ip}$ and $Y_{cp}$ represent the initial and current y-coordinates of the mouse, providing a scroll speed that equals the vertical coordinate distance that the mouse has been dragged. For the map browser application, scrolling speed was computed by:

$$\text{scrollspeed} = \sqrt{(Y_{ip} \Delta Y \text{cp})^2 + (X_{ip} \Delta X \text{cp})^2}$$

Equation 5

providing a scroll speed equal to the distance that the mouse has been dragged in any direction. The zoom level was computed by:

$$\text{zoomlevel} = k \times \text{scrollspeed} \Delta \text{threshold}$$

Equation 6

where $\text{scrollspeed}$ is computed by either Equation 4 or Equation 5. $\text{threshold}$ is the scroll speed (distance from the initial mouse location) up to which no zooming takes place, and $k$ is a constant that controls the rate of change of the zoom level. These are not the equations originally proposed by Igarashi and Hinckley, employing linear transformations of scroll rate and zoom level. However, this mechanism truly bases zooming on scrolling speed. Unfortunately, it is difficult to tell from the information provided, what would be suitable values for $k$ and $\text{threshold}$. However, the minimum zoom level value must be controlled programmatically, with a value of zero providing the maximum level of detail, which is presented when either the mouse is stationary or within the threshold value. The
Cockburn and Savage carried out an evaluation of their implementation for both document and map navigation tasks. For both tasks, the SDAZ systems were compared to standard desktop applications: Adobe Acrobat Reader for document navigation and Paint Shop Pro for map navigation. Navigation was required either short or long distances and was controlled by a mouse for all tasks. Document navigation task completion times using SDAZ were 22% shorter on average in comparison to those for standard scrolling/panning/zooming techniques. In subjective responses participants reported lighter task loads when using SDAZ, and 11 of the 12 subjects expressed a preference for the SDAZ controls. On average, map navigation tasks were completed 43% faster using SDAZ. All subjects preferred the SDAZ interface and reported lower workloads for it.

These results are noticeably better than those reported for SDAZ than Igarashi and Hinckley, which is likely explained by a more responsive implementation, differing task types and participants, and different input devices for the map navigation task. We believe that the rigour of Cockburn and Savage’s study provides concrete evidence of the potential performance gains that can be provided by replacing or augmenting current scrolling techniques with SDAZ.

Neither of the two investigations reported above evaluated this technique in a small screen context, or considered the application of SDAZ on small screen devices. Cockburn and Savage suggest that SDAZ is most suited to use for information spaces of intermediate size, that traditional scrolling is sufficient for small spaces, and that searching methods are required for large spaces (thousands of screens). Their study was carried out on a 19-inch desktop display, with a resolution of 1024x768 pixels, although there is no indication of the dimensions of the windows in which the documents or maps were displayed. Igarashi and Hinckley do not report the attributes of either the display or windows used in their study.

Clearly, display and window dimensions are a critical factor for small screen devices. The definitions of small, intermediate and large information spaces must be amended for application on small screens. For example, 25% of an information space with dimensions of 16x12 inches can be displayed within a window of 8x6 inches on a device such as F in Error! Reference source not found.. By comparison, only 3.6% of the space can be shown on a device such as B. In the next section we discuss our revision of the equations above to explicitly take account of the available display area.

**A revised implementation of SDAZ**

Our implementation addresses a number of factors not discussed in the prior work described above:

1. the proportion of the information space that is currently visible within the window in which it is displayed. This factor supports application on a range of display devices, including small screens, and allows for dynamic reconfiguration of behaviour when the user amends the window size during interaction;
2. how scroll speed maps to horizontal and vertical translations of the information space display;
3. scrolling and zooming behaviour when the bounds of the information space are reached either via scrolling or zooming actions;
4. generalising the algorithms such that they can be used to support navigation both horizontally and vertically, or constrained to one of the two directions.

Further, we amend the implementation so that the zoom level is only amended as the user drags the mouse away from its initial location. This avoids the issue of rapid scaling observed in prior implementations when users reversed their scrolling direction.

**Algorithms**

We follow Igarashi and Hinckley’s revised equations in which scroll speed is linearly dependent upon the current zoom level. In fact we replace the notion of scroll speed with two factors: the required translation of the information space in the x and y dimensions. For both document navigation and map navigation we determine scroll speed via the following algorithm:

```
sdX <- 0
sdY <- 0
if (no horizontal constraint) sdX <- C * visibleArea * scaleValue * dx
if (no vertical constraint) sdY <- C * visibleArea * scaleValue * dy
```
The five conditions in Algorithm 2 provide the following behaviour:

- zooming only occurs as the user drags the mouse away from the location of the start of the zoom/scroll operation
- no further zooming out occurs once either the full width or height of the information space is visible in the window;
- zooming occurs only when the mouse drag distance is within the allowable bounds;
- no further zooming out occurs once the minimum scaling factor has been reached.

The two algorithms are executed only during a pointing action (for example, while a button is depressed on a mouse, or while a stylus is in contact with the screen). When the action ends (the mouse button is released) the information space is transformed in two ways:

- the scale value returns to 1, to display the space at its normal magnification level. This is smoothly animated to minimise visual disruption;
- the information space is repositioned within the viewport to place the location under the pointing device at the end of the operation at the centre of the viewport. Again, this is smoothly animated.
Figure 2: graph of the behaviour of Algorithm 2

The two transformations occur in parallel.

**Implementation**

Our application is implemented in Java 1.4 using the Piccolo\(^1\) 1.0 Zooming User Interface (ZUI) toolkit [1]. It has been run successfully on Microsoft Windows, Linux and Macintosh OS X operating systems.

The SDAZ behaviour of the application is provided by user interface event handler classes. One event handler class (SpeedCoupledPanEventHandler) deals with scrolling operations, and the other (SpeedCoupledZoomEventHandler) with zooming operations. As a result, minimal programming is required to add this behaviour to information space viewing applications.

The application manipulates images of the documents and maps to be navigated. The display image can be a command line argument, as can the minimum and maximum zoom thresholds, the minimum zoom value and navigation direction constraints. The initial layout strategy can also be specified. For textual documents, the viewport of the window can be placed at the start of the document, which is centred horizontally. For images such as maps, the viewport can be placed so that the centre of the map appears at the centre of the window.

Figure XXX illustrates the software in use for viewing a city map. All of the available display area is used to present the information space. Four feedback mechanisms are provided to the user, and are emphasised in the Figure for clarity. They are provided during a navigation action, and removed on completion of the action. At the start of an action, the two concentric circles are placed so that the location of the action on the display it at their centre. The smaller circle indicates the minimum threshold value, and the larger indicates the maximum threshold value, as described above. As the user drags the pointing device, a direction line is drawn between the starting position and its current location, indicating the direction of travel. If the pointer remains within the smaller circle, no scaling of the information space occurs, only scrolling.

The user is free to navigate in any direction. As the pointer moves further away from the starting position, the scroll rate increases. When the pointer moves beyond the inner circle, both scaling and scrolling operations take place. Scaling is progressive until the pointer reaches the outer circle, at which point the minimum scale value is reached and no further scaling occurs, although scrolling is still active.

The rectangle indicates to the user the subset of the information space that will be visible on completion of the operation. It dynamically changes size proportional to the current scale value. To begin with it bounds the entire viewport, and does so until scaling is applied to the information space. It is always centred on the location of the pointing device, and consequently the location under the pointer will be located at the centre of the viewport on completion of the operation.

**Experimental evaluation**

---

We carried out an experiment to compare the efficiency, user perceptions and usability issues of our SDAZ implementation with those of traditional scrolling techniques in the context of small screens.

**Conditions**

Four variables were manipulated in the evaluation:

- *interface type*: either the SDAZ interface or a traditional scrolling/panning interface (henceforth termed ‘standard’);
- *information space type*: either a map or document;
- *task type*: for document navigation the two levels were ‘locate a picture’ and ‘locate a section heading’; for map navigation the levels were ‘locate from direction’ and ‘locate from path’;
- *navigation distance*: the two levels were ‘short’ and ‘long’.

**Subjects**

12 subjects took part in the experiment. Each was an undergraduate or postgraduate university student. 3 of the subjects were female and 9 male. The first language of 10 of the subjects was English, and all subjects had written and spoken English-language proficiency suitable for study at an English-speaking university. All subjects used computers on a daily basis. None had previously used SDAZ interfaces, and 4 had used map viewing software, although mainly via a Web-based interface.

Subjects were recruited through poster and e-mail advertisements, and each received a book token in return for their voluntary participation.

**Tasks**

Each participant completed 48 experimental tasks in two sets. A set of 24 tasks was carried out with each of the two interface types. Half of the subjects carried out SDAZ tasks first, and half standard scrolling interface tasks first. For each interface type, 12 document navigation and 12 map navigation tasks were undertaken. Each set of 12 tasks comprised 3 tasks for each combination of the two task type and navigation distance levels. This is summarised in Table 3. Presentation order for the factor level combinations was counter-balanced to minimise learning effects.
Examples of document navigation tasks are: “Find the next non-colour image down from your starting location” (picture location), and “Find the "Summary" section of the document down from your starting location.” (heading location). Headings were all emphasised within the document text.

Examples of map navigation tasks are: “You are currently located at Wairau Inter School in Coatesville. To the north-west is a school called Coatesville School. Locate it.” (locate by direction), and “You are currently located at St Francis School. Follow highway 16 west until you reach Royal Road School.” (locate by path). All navigation start and end points were the locations of schools, which were coloured yellow on the map, and therefore visually distinct from other types of location.

Figure XXX shows samples of the two information spaces used in the experiment: a road map of a city and surrounding area, and a research paper. Navigation distances were defined as short or long. Short distances were XXX and long distances were XXX.

### Materials

The experiment was conducted using a standard desktop computer with a 1.8Ghz AMD Athlon processor, 240Mb of RAM, and running the Microsoft Windows XP operating system. Display characteristics (resolution and colours) were set to simulate those of a representative target device—a Compaq iPAQ 3870 Pocket PC. Both the SDAZ and standard interface were presented in windows corresponding to the display dimensions of the iPAQ device, namely 2.26 inches wide by 3.02 inches high. Input was via a standard two-button mouse.
The SDAZ interface was as described in Section XXX. The standard interface simulates a tool such as Adobe Acrobat Reader and is illustrated in Figure XXX. It provides the following navigation functions:

- **panning**: when the ‘hand’ tool is selected the user can drag the information space using the pointing device. The extent of the drag distance is constrained by the bounds of the viewport window, and the direction is constrained by the information type. For the map to the left of the figure, panning can occur in any direction, but only vertically for the document to the right;

- **scrolling**: vertical and horizontal scrollbars (vertical only for the document to the right) behave in a normative manner. The arrows at either end allow for small scroll increments which can either be discrete or continuous. The ‘elevators’ within the scrollbars can be dragged backwards and forwards by the user with corresponding scrolling of the viewport. The scrollbar channels can be clicked for a scroll distance equal to the size of the viewport (horizontal or vertical, depending on the scroll direction), and the increments can be either discrete or continuous;

- **zooming**: a zoom level menu (to the top right of the interface) allows selection of discrete zoom levels (25, 50, 75 and 100 percent), and when a value is selected the information space is immediately scaled to the corresponding level. Additionally, zoom-in and zoom-out tools can be used to move between the discrete zoom-levels by clicking on the viewport.

A printed questionnaire was developed and issued to subjects to record their backgrounds and experiences. Printed task sheets were provided to each subject.

**Procedure**
On arrival subjects were presented with a printed Bill of Participants rights, and a consent form describing the nature of the experiment and tasks that they would be requested to perform. Once consent was provided the study proper began.

To begin, a subject was provided with a training workbook for the first class of task to be presented (either document or map navigation). This was divided into two sections, each describing one of the two interaction styles to be used (SDAZ and standard) and associated training tasks. The subject was asked to read the description of the first interaction style, and it was then demonstrated to them by the experimental supervisor. They then worked through four training tasks corresponding to the four combinations of distance (short, long) and task type (locate image or heading for document, locate by direction or path for maps).

The subject was then asked to read the description of the second interaction style, which was then demonstrated to them, followed again by four training tasks. Subjects could ask questions of the supervisor throughout the training period.

Subjects were then provided with an experimental workbook that presented 24 tasks for the first task class (document or map navigation). They were instructed to begin working through the first set of 12 tasks using the first interaction style. Once these were completed, the application software was changed as appropriate and the second set of 12 tasks undertaken.

The subject was offered a short break if required. The same training and experimental task process was then repeated for the second task class.

Upon finishing each task set a subject completed a NASA Task Load Index worksheet, and was prompted to comment on their experiences with the systems they used.

All tasks (including demonstration and training tasks) were presented via an on-screen dialogue window, shown in Figure XXX. Each individual task commenced with the subject clicking the “Read Task” button, which resulted in the display of the task instructions. Once the subject wished to begin the task proper, they clicked the “Start Task” button. This resulted in the display of the required navigation window (SDAZ or scrolling) immediately below the task dialogue, with the view of the document or map placed at the start location of the task. On completing the task to their satisfaction, the subject clicked the “Task Completed” button. The navigation window and the task instructions were then removed. The subject proceeded to the next task by once again clicking the “Read Task” button.
The software was instrumented to automatically record performance data for each task undertaken by each subject. For the SDAZ interface the data recorded for each task was:

- **user interface actions.** The number of distinct user interface actions for the task, with an action defined as a mouse press and mouse release event pairing;
- **action timings.** The start time, end time, and duration of each user interface action recorded in milliseconds;
- **task duration.** The time taken on the task, in milliseconds, defined by the time between the user clicking the “Start Task” and “Task Completed” buttons;
- **accuracy.** The distance, in pixels, of the target location from the centre of the viewport when the subject indicated task completion.

For the standard interface the data recorded for each task was:

- **user interface actions.** The number of distinct user interface actions for the task, with an action defined as any one of: pan, zoom in by clicking, zoom out by clicking, vertical scroll...
action, horizontal scroll action, pan tool selection, zoom out tool selection, zoom in tool selection, zoom level combo-box selection;

• action timings. The start time, end time, and duration of each user interface action recorded in milliseconds;

• task duration. The time taken on the task, in milliseconds, defined by the time between the user clicking the “Start Task” and “Task Completed” buttons;

• accuracy. The distance, in pixels, of the target location from the centre of the viewport when the subject indicated task completion.

For each set of tasks for an interaction style-document type pairing (four sets in total) a subject completed a modified NASA Task Load Index questionnaire. The questions are shown in Figure XXX. Each response was a numeric value of 1, 2, 3, 4 or 5, with the extreme values given indicative textual labels.

The experimental administrator also made notes of critical events and comments made by the subjects.

Results

User interface actions

Table XXX shows the mean number of user interface actions per task for the four combinations of interface and document types. Outlier values of more than 3 standard deviations from the mean have been removed. For the map stimulus, the mean number of actions using SDAZ was 1.97, compared to 17.66 using the standard interface. For the document stimulus, the mean number of actions using SDAZ was 2.76, compared to 6.92 using the standard interface. For both the map and textual document types, the differences are significant (Mann-Whitney, U=1436.5, p<0.0001).

Figure XXX shows the cumulative percentage of tasks against user interface action frequency thresholds for map-based tasks. 55% of the SDAZ-based tasks were completed with only 1 user interface action, and all SDAZ-based tasks were completed with 11 or fewer actions. Only 6% of the map tasks with the standard interface were completed with a single action, and half of them required up to 14 actions to be completed.
Figure XXX shows the corresponding data for text-based tasks. 41% of the SDAZ-based tasks were completed with only 1 interface actions, compared to 23% for the standard interface. All of the SDAZ-based tasks were completed with 13 or fewer interface actions, compared to up to 50 actions for the standard interface.

**Task completion times**

Table XXX shows the mean task completion time (in seconds) for the four combinations of interface and document types. Outlier values of more than 3 standard deviations from the mean have been removed. Mean completion times are longer for the SDAZ interface for both map and textual documents. The difference is not significant (Mann-Whitney, U=9684, p=0.6353) for the map stimulus, but is significant for the document stimulus (Mann-Whitney, U=8183.5, p<0.002).

Figure XXX shows the cumulative percentage of tasks against task completion time thresholds for map-based tasks. For both interface types, approximately 50% of all tasks were completed within 30 seconds, and 80% within 60 seconds. Completion times are more divergent after the 60 second threshold—all standard interface tasks were completed within 145 seconds, yet 5% of the SDAZ tasks took longer than this to be completed.
Figure XXX shows the corresponding data for text-based tasks. Divergence between the performance of the interface types is more pronounced than for the map stimulus. 73% of all tasks were completed within 30 seconds with the standard interface, compared to 52% with SDAZ. The corresponding values at the 60 seconds threshold are 76% and 89% respectively.

**Accuracy**

Accuracy is measured by the distance (in pixels) of the target location from the centre of the viewport on task completion.

Table XXX shows the mean accuracy (in pixels) for the four combinations of interface and document types. SDAZ supported a higher level of accuracy for the map stimulus than the standard interface, and this is a significant difference (Mann-Whitney, U=5988.5, p<0.0001). The opposite is true for the textual stimulus, with the standard interface supporting a significantly higher level of accuracy (Mann-Whitney, U=8430.5, p=0.0061).
For the map stimulus, a distance of less than 140 pixels indicates that the target was definitely visible within the viewport on task completion, and we characterise this as a successful result. The corresponding distance for the textual stimulus is 170 pixels.

For the map stimulus, 92% of all SDAZ tasks and 89% of all standard interface tasks were successful. The respective values for the textual stimulus are 78% and 76%.

Figure XXX shows the frequency distribution of the accuracy on the map stimulus.

Figure XXX shows the frequency distribution of accuracy on the document stimulus.

The descriptive measures of accuracy are somewhat skewed by extreme outlier values. When values greater than one standard deviation from the mean are removed we see the outcomes in Table XXX.

<table>
<thead>
<tr>
<th></th>
<th>Map</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDAZ UI n=144</td>
<td>Standard UI n=144</td>
</tr>
<tr>
<td></td>
<td>Standard UI n=144</td>
<td>SDAZ UI n=144</td>
</tr>
<tr>
<td>mean</td>
<td>88.81</td>
<td>147.07</td>
</tr>
<tr>
<td>sd</td>
<td>280.44</td>
<td>376.64</td>
</tr>
<tr>
<td>median</td>
<td>33.00</td>
<td>64.30</td>
</tr>
</tbody>
</table>

Table XXX: accuracy (in pixels) all data points
Again, the difference is significant between the SDAZ and standard interfaces for both the map stimulus (Mann-Whitney, U=5294, p<0.0001) and the textual stimulus (Mann-Whitney, U=7019.5, p=0.0015).

**Use of tools in the standard interface**

<table>
<thead>
<tr>
<th></th>
<th>SDAZ UI</th>
<th>Standard UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>139</td>
<td>137</td>
</tr>
<tr>
<td>mean</td>
<td>45.97</td>
<td>70.69</td>
</tr>
<tr>
<td>sd</td>
<td>38.19</td>
<td>36.91</td>
</tr>
<tr>
<td>median</td>
<td>30.90</td>
<td>62.71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SDAZ UI</th>
<th>Standard UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>134</td>
<td>135</td>
</tr>
<tr>
<td>mean</td>
<td>153.56</td>
<td>152.00</td>
</tr>
<tr>
<td>sd</td>
<td>55.65</td>
<td>47.36</td>
</tr>
<tr>
<td>median</td>
<td>136.50</td>
<td>145.90</td>
</tr>
</tbody>
</table>

**Table XXX: accuracy data with outliers removed**

**Analysis of each interface by task type (maps)**

For the map stimulus subjects carried out four types of navigation task: long distance-directional (LD); short distance-directional (SD); long distance-following path (LF); and short distance-following path (SF). Table XXX shows the performance measures for each task type, with outliers removed as described above.
User interface actions (maps)

A Kruskal-Wallis one-way analysis of variance by ranks reveals no significant difference between task types for the SDAZ interface (KW=0.9121, p=0.8225). There are significant differences for the standard interface (KW=16.4, p=0.0009). Dunn’s multiple comparisons post-test shows there to be significantly more actions for the LD tasks than for SD (p<0.05) and SF (p<0.01).

Time (maps)

There is a significant difference between at least one pair of task types (KW=11.14, p=0.011) for the SDAZ interface. Dunn’s post-test shows the LD tasks to have taken significantly longer than the SD tasks (p<0.05) and SF tasks (p<0.05).

There are also significant differences for the standard interface (KW=21.98, p<0.001). The LD tasks were completed significantly slower than SD tasks (Dunn, p<0.01), and SF tasks (Dunn, p<0.001).

Distance (maps)

There is a significant difference between task types for the SDAZ interface (KW=13.71, p=0.003). SD tasks were completed significantly more accurately than LD tasks (Dunn, p<0.01). There were no significant differences between task types for the standard interface (KW=0.9284).
Analysis of each interface by task type (textual)

For the textual stimulus subjects carried out four types of navigation task: long distance-locate heading (LH); short distance-locate heading (SH); long distance-locate picture (LP); and short distance-locate picture (SP). Table XXX shows the performance measures for each task type.

User interface actions (textual)

A Kruskal-Wallis one-way analysis of variance by ranks reveals a significant difference between task types for the SDAZ interface (KW=53.02, p<0.0001). All pair-wise differences are significant (p<0.05) except for SH and SP for which there is no significant difference. There are no significant differences present between task types for the standard interface.

Time (textual)

For the SDAZ interface there are significant differences between task types (KW=80.94, p<0.0001). These are between LH and both SH (Dunn, p<0.001) and SP (Dunn, p<0.001), and between LP and both SH Dunn, (p<0.001) and SP (Dunn, p<0.001).

There are also significant differences present for the standard interface (KW=73.18, p<0.0001). The are between LH and each of the other task types (Dunn, p<0.001 for each), and between LP and SP task types (Dunn, p<0.001).

Distance (textual)

There is no significant difference between task types for the SDAZ interface (KW=1.37, p=0.712). For the standard interface a significant difference (KW=9.48, p=0.235) is found between the SH and SP task types Dunn, (p<0.05).

Analysis between interfaces by task type (maps)

User interface actions

The number of actions taken to complete tasks differed significantly between the two interfaces for all task types (LD, LF, SD, SF) with p<0.0001 (Mann-Whitney).

Time

There was no significant difference between task completion times for the two interfaces for any of the task types, with p>0.55 in all cases (Mann-Whitney).

Distance

There was no significant difference between accuracy on LD tasks between the two interfaces (Mann-Whitney, p=0.3536). The difference for all other task types was significant (Mann-Whitney, p<0.0001).

<table>
<thead>
<tr>
<th>Task Type</th>
<th>SDAZ Mean</th>
<th>SDAZ SD</th>
<th>SDAZ Mean</th>
<th>SDAZ SD</th>
<th>Standard Mean</th>
<th>Standard SD</th>
<th>Standard Mean</th>
<th>Standard SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>LH</td>
<td>2.9</td>
<td>1.9</td>
<td>16.2</td>
<td>24.8</td>
<td>78.3</td>
<td>45.2</td>
<td>64.9</td>
<td>41.3</td>
</tr>
<tr>
<td>LP</td>
<td>4.5</td>
<td>2.9</td>
<td>7.7</td>
<td>8.8</td>
<td>46.0</td>
<td>23.1</td>
<td>20.5</td>
<td>9.4</td>
</tr>
<tr>
<td>SH</td>
<td>1.7</td>
<td>1.4</td>
<td>3.3</td>
<td>4.0</td>
<td>14.4</td>
<td>10.8</td>
<td>16.7</td>
<td>11.5</td>
</tr>
<tr>
<td>SP</td>
<td>1.6</td>
<td>1.6</td>
<td>3.2</td>
<td>2.7</td>
<td>17.7</td>
<td>22.0</td>
<td>10.1</td>
<td>6.3</td>
</tr>
</tbody>
</table>

Table XXX: measures by task type for documents
Analysis between interfaces by task type (textual)

User interface actions
There was a significant difference in the number of user interface actions required in the interfaces for SP tasks (Mann-Whitney, p=0.0052). There was no difference for the other task types with p>0.25 in all cases.

Time
There was a significant difference between the interfaces for task completion time for LP tasks only (Mann-Whitney, p<0.0001). There was no difference for the other task types with p>0.105 in all cases.

Distance
There was a significant accuracy difference between interfaces for SH tasks only (Mann-Whitney, p=0.002). The difference for LH tasks was not quite significant (Mann-Whitney,p=0.0842), and for LP and SP tasks there was no significant difference with p>0.23 (Mann-Whitney) in both cases.

Subjective workload measures
A task load questionnaire, based on the NASA Task Load Index, was administered to each subject after each set of tasks for a document type-interface type pairing. Responses were on a scale of 1 to 5, and were normalised so that lower values reflect lower task loads. Table XXX shows mean ratings for each interface for both maps and textual documents, with the lower of each pair of values emphasised. The overall mean ratings are almost identical, with values of 2.70 and 2.73 for the SDAZ and standard interfaces respectively for map documents. The corresponding values for textual documents are 2.60 and 2.61.

There is no significant difference between the task load responses for the two interface types for map documents (Wilcoxon matched-pairs signed ranks test, p=0.8279), or for textual documents (Wilcoxon matched-pairs, p=0.8593).

Discussion

Future work

Control feedback mechanisms
Both Cockburn and Savage and Igarashi and Hinckley provide velocity feedback scrollbars in their document navigation interfaces. The velocity bar is detailed in Figure 3(a), and Figure 3(b) shows the arrangement of bar and document in their implementations. The intention of the bar is to provide users with feedback as to their scroll direction, whether or not zooming is taking place, and the available

<table>
<thead>
<tr>
<th></th>
<th>maps</th>
<th>textual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sdaz</td>
<td>standard</td>
</tr>
<tr>
<td>mental and perceptual activity</td>
<td>3.3</td>
<td><strong>3.0</strong></td>
</tr>
<tr>
<td>mental demand</td>
<td>2.7</td>
<td>2.7</td>
</tr>
<tr>
<td>complexity</td>
<td>2.8</td>
<td><strong>2.5</strong></td>
</tr>
<tr>
<td>forgiveness</td>
<td><strong>3.5</strong></td>
<td>3.7</td>
</tr>
<tr>
<td>physical activity</td>
<td><strong>2.4</strong></td>
<td>3.2</td>
</tr>
<tr>
<td>physical demand</td>
<td><strong>1.8</strong></td>
<td>2.3</td>
</tr>
<tr>
<td>perceived success</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>satisfaction with performance</td>
<td>2.7</td>
<td><strong>2.3</strong></td>
</tr>
<tr>
<td>work level</td>
<td>3.2</td>
<td><strong>2.8</strong></td>
</tr>
<tr>
<td>frustration</td>
<td><strong>2.8</strong></td>
<td><strong>2.6</strong></td>
</tr>
<tr>
<td>mean rating</td>
<td>2.70</td>
<td>2.73</td>
</tr>
</tbody>
</table>

Table XXX: task loading responses
increase in scroll speed. Scroll direction and zooming activity should be evident from transformations on the information space. However, the zoom threshold is not, and this feedback allows users to control zoom activation. However, this approach suffers from the same disadvantage as conventional scrollbars—the user’s focus of attention needs to shift between the velocity bar and the information space.

To ameliorate this problem we have implemented an alternative feedback mechanism (shown earlier in Figure XXX), which incorporates circular threshold cues and a target-bounding rectangle. This mechanism places the requisite feedback at the point at which the user is most likely to be focussing their attention, whilst minimally obscuring components of the information space. We are currently in the process of experimentally evaluating the efficacy of this mechanism.

**Navigational feedback mechanisms**

Outlying values in the data (particularly with respect to time and accuracy), and observations during the study suggest that when subjects failed at tasks, they failed very badly. Through observation, the main cause of these extreme failures is the fact that the subjects became ‘lost’—they did not know how to reach the target, or more importantly, how to return to their starting location to reorient themselves and begin the task again. On large displays, with surrounding context for the user’s focus, this problem is likely to be less evident. On small displays, minimal navigational activity can remove the user’s starting point from the viewport.

A common solution to support the user in understanding how the area visible in the viewport relates to the whole information space, is to provide an overview map. Such an overview would condense the entire space into a small display area, with the viewport location emphasises. Although perhaps feasible on a desktop display, small displays have limited space to present an overview without obscuring the detailed view of the information space.

Our solution is to apply the ‘halo’ technique (REF XXX), which we have already implemented and will evaluate in the near future. Figure XXX illustrates the use of halos. The image to the left of the figure shows the user in the course of a navigation action on a London Underground map. To the bottom right a halo indicates the both the direction and distance to be travelled to return to the start location of the navigation action. The halo is always situated at the edge of the viewport, and its position is dynamically updated during navigation. By moving in the direction of the halo the user moves towards the start location. The width of the halo reflects the distance that the user needs to travel to return to the start location—the wider the halo, the greater the distance. By making halos semi-transparent, and peripheral to the user’s focus within the viewport no additional display space is required, and no detail of the map is obscured.
The use of halos can be extended for situations where the user has identified distinct targets to which they wish to navigate. To the right of Figure XXX, the user has indicated that they wish to locate ‘Archway’ and ‘North Greenwich’ stations. A halo is added for each target, behaving in the same way as the halo indicating the start location, although discriminated from it by colour.